Purdue University
Purdue e-Pubs

IWRRC Technical Reports | Indiana Water Resources Research Center

2-1-1979

Multilevel Approach To Urban Water Resources Systems Analysis-Application To Medium Size Communities: Interactive Multiple Objective Optimization

K. J. Musselman

J. J. Talavage

Follow this and additional works at: http://docs.lib.purdue.edu/watertech

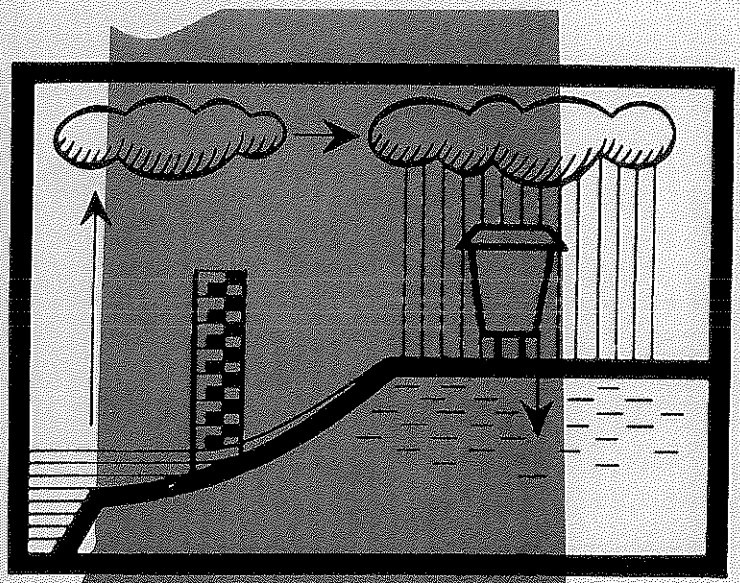
Musselman, K. J. and Talavage, J. J., "Multilevel Approach To Urban Water Resources Systems Analysis-Application To Medium Size Communities: Interactive Multiple Objective Optimization" (1979). *IWRRC Technical Reports.* Paper 121.
http://docs.lib.purdue.edu/watertech/121

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

TECHNICAL REPORT NO. 121

Multilevel Approach to Urban Water Resources
Systems Analysis-Application to Medium Size Communities

INTERACTIVE MULTIPLE OBJECTIVE OPTIMIZATION



by

Kenneth J. Musselman

Joseph J. Talavage

February 1979

PURDUE UNIVERSITY
WATER RESOURCES RESEARCH CENTER
WEST LAFAYETTE, INDIANA

Water Resources Research Center
Purdue University
West Lafayette, Indiana 47907

INTERACTIVE MULTIPLE OBJECTIVE OPTIMIZATION

by

Kenneth J. Musselman
and
Joseph J. Talavage

The work upon which this report is based was supported in part by funds provided by the United States Department of the Interior, Office of Water Research and Technology, as authorized by the Water Resources Research Act of 1974 (PL88-379) as amended.

Period of Investigation: January 1977 - August 1979
Partial Completion Report for OWRT-B-083-IND
Matching Fund Agreement 14-31-0001-5213

Purdue University Water Resources Research Center
Technical Report No. 121
February 1979

ACKNOWLEDGEMENTS

The present report covers part of the work done under Matching Grant Fund OWRT-B-083-IND and was supported in part by the Office of Water Research and Technology, Department of the Interior, and in part by Purdue University. The authors wish to express their appreciation to Dr. Dan Wiersma, Director of the Purdue University Water Resources Research Center, to Dr. John F. McLaughlin, Head of the School of Civil Engineering and to Dr. Wilbur L. Meier, Jr., Head of the School of Industrial Engineering at Purdue University for their assistance in the administration of the project.

The present report is a slightly condensed version of the Doctoral Thesis "An Interactive Tradeoff Cutting Plane Approach to Continuous and Discrete Multiple Objective Optimization" submitted by K. Musselman to the Graduate Faculty at Purdue University in December, 1978, which may be obtained from University Microfilms, Ann Arbor, Michigan. The work was performed under the supervision of Professor J. J. Talavage.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABSTRACT	x
CHAPTER I - CONCEPTS AND TERMINOLOGY	1
Definition of the Problem	2
Preliminary Definitions and Assumptions	4
Tradeoffs	9
Methods of Solution	12
The Problem to be Studies	14
Additional Assumptions	15
Superiority Test	17
A Usable Direction	18
Algorithmic Convergence	19
Organization of the Report	23
CHAPTER II - THE GEOFFRION-DYER-FEINBERG INTERACTIVE ALGORITHM	25
General Overview	25
Description	26
The Interactive Frank-Wolfe Algorithm	29
Significant Attributes	29
Some Limitations of the GDF Algorithm	30
CHAPTER III - THE TRADEOFF CUTTING PLANE ALGORITHM	33
Intent of the Tradeoff Cutting Plane Algorithm	33
Concepts Behind a Cut	36
Locating the Next Tradeoff Point	41
Tradeoff Cutting Plane Algorithm	45
Proof of Convergence	53
Tradeoff Sensitivity	68
CHAPTER IV - STORM DRAINAGE - AN APPLICATION	76
Overview	76
Original Formulation	80
Multiple Objective Formulation	80
Problem Statement	83
Tradeoff Policy A	89

TABLE OF CONTENTS (Continued)

Page

Tradeoff Policy B 94
Tradeoff Policy C 101
Tradeoff Policy D 101
Discussion 106

CHAPTER V - DISCRETE OPTIMIZATION VIA THE TRADEOFF CUTTING PLANE ALGORITHM 110

General Approach 110
Development of the Algorithm 111
Ancillary Definitions 118
Discrete Version of the TCP Algorithm 121
Finiteness of the Algorithm 123
Results Relating to the Final Potential Set 126
Modifications 134
Examples 141

CHAPTER VI - SUMMARY AND RECOMMENDATIONS 153

Overview 153
Areas for Future Research 154

BIBLIOGRAPHY 159

APPENDICES

Appendix A: An Amalgam of Definitions of Lemmas 164
Appendix B: Method of Centers 172
Appendix C: Program Listing 177

LIST OF TABLES

Table		Page
4.1	Analytical Model of Subbasin 13's Storm Drainage Problem	82
4.2	Regression Models of the Simulation Output Parameters	84
4.3	Bounds Associated with Constraints g_1 through g_{35}	88
4.4	Sequence of Points Generated from SP1 under Policy A	91
4.5	Sequence of Points Generated from SP2 under Policy A	92
4.6	Sequence of Points Generated from SP3 under Policy A	93
4.7	Sequence of Points Generated from SP1 under Policy B	102
4.8	Sequence of Points Generated from SP2 under Policy B	103
4.9	Sequence of Points Generated from SP3 under Policy B	104
4.10	Sequence of Points Generated under Policy C	105
4.11	Sequence of Points Generated under Policy D	107
Appendix Table		
C.1	FORTRAN Representation of Terms	178

LIST OF FIGURES

Figure		Page
1.2	Objective Space	10
3.1	Decision Space	39
3.2	Objective Space	39
3.3	Translated Cut in Decision Space	40
3.4	Flowchart of the TCP Algorithm	48
3.5	Tradeoff Sensitivity in the Case of Two Objectives	72
3.6	Tradeoff Sensitivity Printout	73
4.1	Conceptualization of Subbasin 13's Storm Drainage Problem	78
4.2	Response Surface and Regression Model of the Expected Economic Loss Due to Flooding	85
4.3	Response Surface and Regression Model of the Average Number of Pounds per Year of Suspended Solids	86
4.4	Total Cost of the Drainage System	90
4.5	Drainage System Cost as a Function of the Iteration Number	95
4.6	Drainage Network Cost Under Policy B	96
4.7	Storage Facility Cost Under Policy B	97
4.8	Treatment Facility Cost Under Policy B	98
4.9	Expected Flood Damage Cost Under Policy B	99
4.10	Expected Economic Loss Due to Flooding Under Policy B	100

LIST OF FIGURES (Continued)

Figure		Page
5.1	Tradeoff Cut Elimination Scheme	114
5.2	Inability to Locate the Best-Compromise Solution ..	116
5.3	The Plausible Set Ω = { B,C }	120
5.4	Flowchart of the TCP Algorithm (Discrete Version)	124
5.5	Variation in the Final Potential Set	135
5.6	Pseudocut Illustration	137
5.7	Hypercube Caveat	140
5.8	Tradeoff Cuts Connected with the Discrete Version of the TCP Algorithm	144
5.9	First Iteration Pseudocut	147
5.10	Tradeoff Cuts Connected with the Pseudocut Routine	149
5.11	First Iteration of the Hypercube Routine	150
5.12	Second Iteration of the Hypercube Routine	152
Appendix Table		
B.1	Locating $\underline{z}^{i+1}$ from $\underline{z}^{i}$	176

ABSTRACT

Decision problems often arise in which several, non-commensurate and conflicting objectives must be considered simultaneously. The subject of this report is concerned with first investigating a general nonlinear class of these problems and then developing a means of solving problems found in this class. The results of this research are divided into four related areas:

(1) identifying properties and relationships which exist within this class of problems and which prove useful to their resolution,

(2) developing a user-oriented algorithm, using these properties, to solve nonlinear multiple objective optimization problems,

(3) applying the algorithm to a representative problem, and

(4) modifying the algorithm to address discrete nonlinear multiple objective optimization problems.

In identifying the properties and relationships which exist between the decision space and the objective space, insight is gained into how the decision maker can avoid considering alternatives which are inferior to his present state. It is shown, by means of a "tradeoff cut", how the feasible region in decision space is able to be reduced without jeopardizing the final (best-compromise) solution.

The concept of a tradeoff cut is fundamental to the optimization technique developed in this dissertation. This optimization technique, termed the Tradeoff Cutting Plane algorithm, uses these cuts to reformulate the original multiple objective problem into a series of pairwise decisions. By repeatedly locating a feasible solution and then modifying the problem by means of these tradeoff cuts, one converges to the best-compromise solution. The algorithm makes prudent use of the decision maker by closely matching his ability to supply the information necessary to unify the various objectives. Furthermore, upon termination, the algorithm conducts a sensitivity analysis of the decision maker's inputs supplied during the solution process.

A decision problem concerned with the expected storm drainage needs of an urban subbasin is analyzed by means of this algorithm. Specifically, the problem involves minimizing various pollutant loads and costs by adjusting the drainage system's local detention storage capacity, maximum treatment rate, and maximum allowable overflow rate. Convergence results are shown for several tradeoff policies.

Finally, the Tradeoff Cutting Plane algorithm is modified to resolve the question of how to find integer solutions to these nonlinear multiple objective optimization problems. The peculiar difficulties involved with directly applying the continuous version

of the algorithm are discussed in detail. It is also explained how the modified version of the algorithm is able to circumvent these difficulties.

CHAPTER I

CONCEPTS AND TERMINOLOGY

Decision makers, be they individuals or collective bodies, are frequently beset with having to find solutions to large-scale problems. These problems, found both in the public and private sector, have become more and more complex due to the active participation of public-interest groups, legislative bodies, and the public media. This widening of the participating community creates the incentive to individually represent and treat those factors relevant to the various interest groups. As a consequence, these problems become characterized by multiple, non-commensurate, and often conflicting objectives. For example, water resource projects must often be concerned about flood control, water quality, land use, erosion and sedimentation, fish and wildlife, outdoor recreation, flooding and drainage, water supply, water and land policy, and so forth. As a result, decision making has become increasingly more difficult.

Being inundated with such a vast amount of information, the decision maker often chooses to either ignore or oversimplify the original problem [13]. He begins to disregard relevant considerations in order to avoid the multiple objective aspect of the problem. So as the number of objectives increases, the ability of the decision maker to comprehend and correctly solve the problem decreases. With the volume of information exceeding his ability to process it, his own personal values are

inclined to suppress those of the organization, be it a public committee, a private firm, a government group, a university, or some similar body. One can easily recognize that this type of an approach has a tendency to serve the needs of the decision maker more than the needs of the organization.

How then should one systematically analyze these complex problems?

A great deal of effort has been and is now being devoted to the development of various methods to assist the decision maker in these situations. Techniques range from those which consider only one objective at a time to those which look simultaneously at all the objectives. A discussion of these methods is deferred until the problem and the types of solutions associated with it are more clearly defined. Nonetheless, it can be stated that these methods all share the common goal of formulating a means to consider more than one objective in a way that is theoretically sound and operationally efficient.

1.1. Definition of the Problem

The type of problem we have been discussing is generally termed a multiple objective optimization problem. It concerns the solution of mathematical programming models with more than a single objective. Simply stated, the object is to maximize r ($r > 1$) distinct objective functions defined over the set of feasible decisions. Numbering the objectives from 1 to r reflects no relative importance among them. In mathematical terms the problem is defined to be:

$$\text{maximize} \quad \underline{f}(\underline{x}) \equiv (f_1(\underline{x}), f_2(\underline{x}), \ldots, f_r(\underline{x})) \qquad (1.1)$$

subject to

$$g_j(\underline{x}) \leq 0, \qquad j=1,2,\ldots,m \qquad (1.2)$$

where $\underline{x}$ is an n-dimensional vector of decision variables.

$f_i(\underline{x})$, $i=1,2,\ldots,r$, are r distinct objective functions.

$g_j(\underline{x})$, $j=1,2,\ldots,m$, are m constraint functions.

Since equations can be replaced by two inequalities in opposite directions, equality constraints will not be explicitly represented.

Although many writers prefer to distinguish among attributes, criteria, objectives and goals, this author will not hold to any distinction. Furthermore, when examining complex problems the objectives are occasionally subjective in nature and are not known with any certainty. Here, however, it is assumed that the objectives are precise analytical expressions which accurately reflect a decision's worth or value. Associated with any point in decision space there is a certain known consequence in objective space. The analysis is assumed to start at a stage where the decision situation has been explicitly defined -- the various alternatives have been identified and are able to be evaluated against known objectives.

Throughout this analysis it is assumed that there is a well-specified, identifiable, unitary decision maker. This is not to deny that many problems are incapable of being resolved in this manner. One need not search very far to find numerous examples where the decision involves an intricate composite of opposing individual viewpoints. Approaches which ostensibly recognize multiple decision makers are discussed by Sen [50], Fishburn [18], Haimes and Hall [25], and Keeney

and Raiffa [34]. However, at some point in most decision processes, a single decision maker (or a group of individuals that may be viewed as a single decision maker) is called upon to make a choice. Thus, limiting the scope of this report to only those situations involving single decision makers is not excessively restrictive.

1.2. Preliminary Definitions and Assumptions

First, attention is focused on what is termed the feasible region of the decision maker's "decision space".

Definition. (Feasible Region in Decision Space)

The region defined by the constraint set (1.2) in n-dimensional Euclidean vector space,

$$X = \{\underline{x} \mid g_j(\underline{x}) \leq 0, \quad \forall_j \}$$

will be referred to as the feasible region in the decision space.

The r-dimensional objective function maps the feasible region in decision space X into the feasible region in objective space $\underline{f}(X)$, defined on the r-dimensional Euclidean vector space.

Definition. (Feasible Region in Objective Space)

The set

$$Y = \{\underline{y} \mid \underline{y} = \underline{f}(\underline{x}), \ \underline{x} \in X\}$$

will be referred to as the feasible region in the objective space.

Ultimately, the decision maker seeks a course of action represented by some $\underline{x}$ in X. However, it will be the feasible region Y which will be of most immediate concern to him for it is the points in Y which serve to indicate the levels of the objectives.

Since it is ambiguous to speak of optimizing r objective functions, it is important to clarify what is meant by an optimal solution and further explain the type of solution one hopes to reach.

Definition. (Superior Solution)

A superior (optimal) solution is one which maximizes all the objectives simultaneously; a solution $\underline{x}^{o}$ to problem (1.1)-(1.2) is said to be superior if and only if $\underline{x}^{o} \in X$ and $\underline{f}(\underline{x}^{o}) \geq \underline{f}(\underline{x})$ for all $\underline{x} \in X$.

It often arises, however, that objectives will conflict with each other in that improvement in any one objective is done at the expense of some other objective. The decision maker is faced with the problem of having to decide how much to sacrifice of one objective in order to gain in another. It becomes a problem involving value tradeoffs among the objectives. Later, it will be shown how one is able to constructively use these tradeoffs to resolve this problem of conflicting objectives.

It is assumed that the decision maker prefers higher values of all the objectives to lower values. If an objective is used in which lower values are preferred to higher values, then the corresponding objective function is defined as its negative. In this way, higher values will always be the preferred values. Also, the relative preferences that the decision maker has for points in X are based entirely upon their respective functional values in Y. The following definition will help clarify this last statement.

Definition. (Preferred)

If $\underline{f}(\underline{x}^1)$, $\underline{f}(\underline{x}^2) \in Y$, $\underline{f}(\underline{x}^1) \geq \underline{f}(\underline{x}^2)$, and $f_i(\underline{x}^1) > f_i(\underline{x}^2)$ for some $i=1,2,\ldots,r$, then $\underline{x}^1$ is preferred to $\underline{x}^2$.

For this preference relation to hold, it is assumed that each objective is ordinally independent of the others. This means that if

$$(f_i, \overline{f}_i) \geq (f_i^*, \overline{f}_i),$$

then

$$(f_i, \overline{f}_i') \geq (f_i^*, \overline{f}_i'),$$

where $\overline{f}_i$ is the complementary set of objectives with respect to f_i, and $\overline{f}_i'$ represents a different set of objective values than $\overline{f}_i$.

From the above definitions and assumptions it becomes clear that the decision maker should restrict his attention to what are called efficient solutions.

Definition. (Efficient Solution)

An efficient (Pareto-optimal, noninferior, nondominate, admissible) solution is a feasible solution, $\underline{x}^* \in X$, for which there does not exist any other feasible solution, $\underline{x} \in X$, such that $\underline{f}(\underline{x}^*) \leq \underline{f}(\underline{x})$ and $f_i(\underline{x}^*) < f_i(\underline{x})$ for some $i=1,2,\ldots,r$.

In other words, an efficient solution is a feasible solution for which there exists no other feasible solution which is preferred to it.

Clearly, an efficient solution is, in general, not unique. The collection of all efficient solutions, X^*, is a subset of the feasible

region in the decision space X. The objective function $\underline{f}$ maps these efficient solutions into what is referred to as the efficient frontier $\underline{f}(X^*)$. Note that the efficient frontier is defined in the objective space and $\underline{f}(X^*) \subset \underline{f}(X)$.

Without further preference information regarding the objective functions f_i, a complete ordering of the feasible solutions cannot be obtained. A complete ordering is possible, however, if the decision maker is willing to indicate the points in objective space for which he is indifferent. With this added preference information, a continuous, real-valued function U can be defined on this space for which $U(\underline{f}^1) = U(\underline{f}^2)$ if and only if $\underline{f}^1$ is indifferent to $\underline{f}^2$ [32]. The function U is called an overall preference function.

It has already been assumed that the decision maker prefers higher values of any objective. This assumption is termed non-satiation. Assuming U is differentiable, this means that all 1^{st} order partial derivatives of the overall preference function U are positive. That is,

$$\frac{\partial U}{\partial \underline{f}}\,(\underline{f}(\underline{x})) \;=\; \left(\frac{\partial U}{\partial f_1}\,(\underline{f}(\underline{x})), \ldots, \frac{\partial U}{\partial f_r}\,(\underline{f}(\underline{x}))\right) \;> 0.$$

Another assumption required of the overall preference function is for it to be concave. Usually, to guarantee the concavity of U on X, one of the following two conditions are assumed to exist: either (i) U is concave on Y and each f_i is linear on X, or (ii) U is concave increasing on Y and each f_i is concave on X. Since condition (ii) is the situation commonly encountered in practice (i.e., non-satiation), it will be the one assumed in this analysis.

These restrictions on U are primarily made in order to simplify discussion and lend some exactness to the subject. They do not normally impose any significant practical limitations on the decision maker.

Previously, it was shown that a decision maker should focus his attention of efficient solutions. However, without additional preference information regarding the objective space, one particular efficient solution cannot be considered preferable to any other. In the presence of the overall preference function U, the "best" efficient solution can now be identified.

Although some authors choose to use the term "optimal" or "preferred" for this "best" solution, this author will use the phrase "a best-compromise solution". Originally coined by Belenson and Kapur [3], it alludes to the fact that such a solution is "best" only in terms of the particular preference function used.

Definition. (Best-Compromise Solution)

A best-compromise solution is an efficient solution which is chosen as the final decision based on the decision maker's overall preference function U.

By fixing U at some constant value a, an indifference curve (for $U = a$) can be established between any two objectives. For example, the indifference curve between f_2 and f_4 is constructed by varying f_2 and f_4 while holding U and the other objective functions (i.e., $f_1, f_3, f_5, f_6, \ldots, f_r$) fixed. The resultant indifference curve, which is in two-dimensional space, has certain properties associated with it. Taken from Baumol [1], they are as follows:

P1) An indifference curve, which lies above and to the right of another, represents preferred combinations of the objectives.

P2) Indifference curves have a negative slope.

P3) Indifference curves can never meet or intersect, so that only one indifference curve will pass through any one point (in the objective space).

P4) The absolute slope of an indifference curve diminishes toward the right. Consequently, an indifference curve is convex to the origin.

Figures 1.1 and 1.2 illustrate some of the concepts defined to this point. An example of a feasible region in a two-dimensional decision space is shown in Figure 1.1. Also shown in the figure is the best-compromise solution. Figure 1.2 pictures the corresponding feasible region in objective space, the feasible region Y, the efficient frontier, indifference curves, and the best-compromise.

1.3. Tradeoffs

Indifference curves are used to depict the tradeoffs which exist between two objectives. The negative slope of the tangent to these indifference curves is termed the marginal rate of substitution or tradeoff ratio.

Definition. (Tradeoff Ratio)

The tradeoff ratio (marginal rate of substitution) between objectives f_1 and f_2 at a point $\underline{f}(\underline{x}^o)$ in Y is the ratio

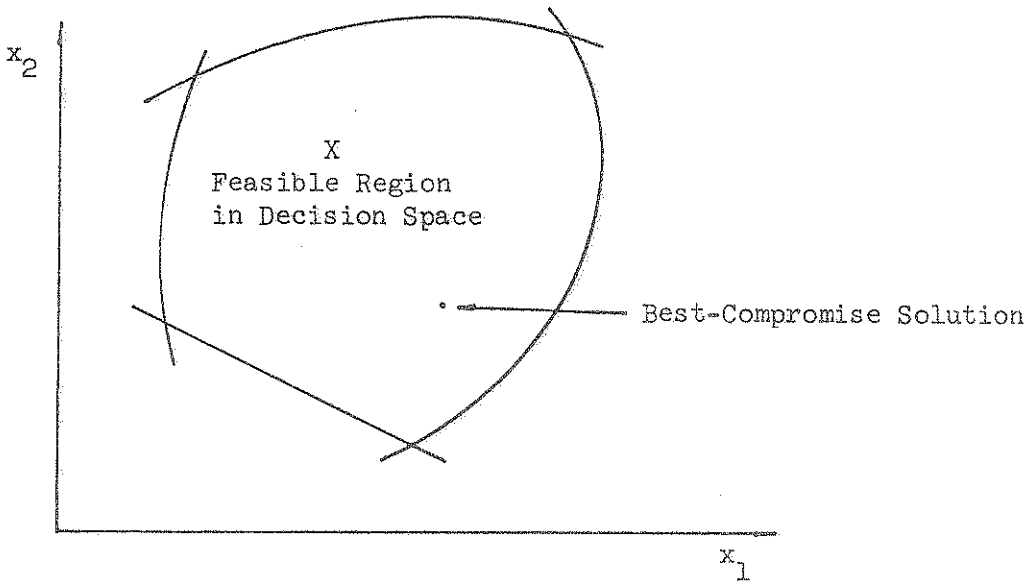


Figure 1.1 Decision Space

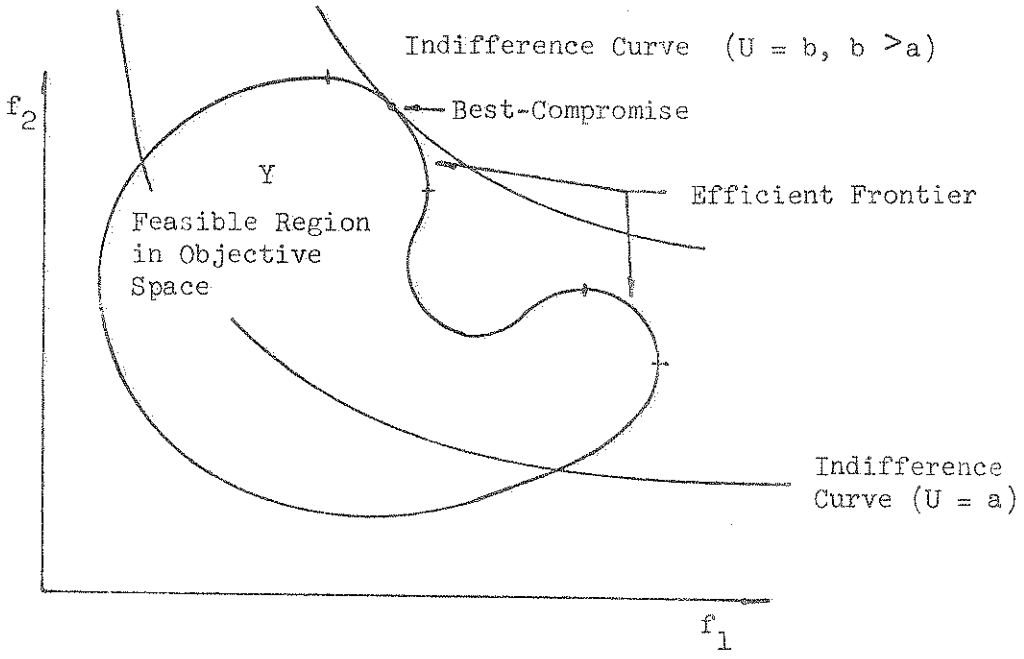


Figure 1.2 Objective Space

$$\frac{(\partial U/\partial f_2) \mid \underline{f}(\underline{x}) = \underline{f}(\underline{x}^o)}{(\partial U/\partial f_1) \mid \underline{f}(\underline{x}) = \underline{f}(\underline{x}^o)}.$$

To further illustrate, a tradeoff ratio of r is the marginal rate of substitution of objective f_1 for f_2. It means that at the current point $\underline{f}(\underline{x}^o)$ the decision maker is indifferent to a decrement of r units of f_1 provided f_2 is incremented by one unit. This ratio will, in general, depend on the levels of both f_1 and f_2. It might also depend on the levels of the other objective functions.

Independence between the (conditional) indifference curves of two objectives and the levels of the other objectives is termed preferential independence [53]. It implies that the indifference curves of any two objectives are the same regardless of the other functional values. In this paper only local information about the overall preference function U is needed. As a result, the restrictive assumption of preferential independence is averted.

Throughout this thesis it will be assumed that the decision maker can provide tradeoffs between the various objective functions. Since these local changes which preserve indifference can be made around any point in objective space, the objectives are considered compensatory. Accordingly, one generally refers to this as a compensatory tradeoff model. In contrast to this are the noncompensatory models which include lexicographic ranking schemes reviewed by Fishman [17], elimination by aspects discussed in Tversky [54, 55], and several others described in MacCrimmon [38].

1.4. Methods of Solution

Haimes et. al. [26] point out that basically the methods used in identifying the best-compromise solution fall into two categories: 1) those which identify efficient solutions before trying to determine the best-compromise solution and 2) those which endeavor to directly solve for it. Cohon and Marks [10] have further subdivided this latter category into methods which rely on "prior articulation of preferences" and those which rely on "progressive articulation of preferences".

Techniques which rely on the prior articulation of preferences include Goal Programming [9, 31, 36], Optimal Weighting [39, 40], ELECTRE [6, 48], and the Surrogate Worth Tradeoff Method [24, 26]. The decision maker arrives at a complete ordering or at least a "more complete" ordering (as in the case of ELECTRE) by the use of these techniques. Preferences laid down by the decision maker prior to the start of the problem act as a basis for these orderings.

The techniques which rely on the progressive articulation of preferences follow a general pattern. First, they locate a feasible (and perhaps efficient) solution. Next, they modify the problem according to the decision maker's preferences. By repeating this general procedure, the decision maker either converges towards the best-compromise solution, or he terminates the process prior to reaching this limiting point.

The objective of this approach is to yield a satisfactory decision after a reasonable number of iterations and within a reasonable amount of time [52]. This final decision should be one in which the decision

maker is relatively convinced that no other significantly improved decision exists.

Methods representative of this approach include Step Method [4], Sequential Multiobjective Problem Solving [41], Iterative Weighting Method [35], Progressive Orientation Procedure [5], Zionts-Wallenius Interactive Programming Method [64], and the Geoffrion-Dyer-Feinberg Interactive (GDF) algorithm [21, 22].

More recently, Oppenheimer [43] has attempted to blend these two approaches (viz., prior and progressive articulation of preferences) by using a global preference function in conjunction with the Geoffrion-Dyer-Feinberg algorithm. A prior analysis is conducted to determine what functional form best approximates the decision maker's overall preference function. Local tradeoff information then progressively updates this function as one approaches the best-compromise solution.

A major difficulty with Oppenheimer's "proxy" approach is the arduous, confusing and very involved prior analysis which is required. Once this analysis has been completed, however, the process has been shown to progress extremely well. Essentially, the question is whether or not this involved prior analysis is worth the improvement one gains in terms of the rate of convergence.

To date, very little attention has been paid to discrete multiple objective mathematical programming problems. Only recently have there been articles treating this important area. Zionts [63] has suggested an interactive approach based on his and Wallenius' interactive programming method, Lee [37] and Ignizio [31] have extended the goal programming simplex method to integer and zero-one goal programming

problems, and Bitran [7] has developed some theoretical results for the zero-one linear multiple objective case.

The author's interest lies primarily with the Geoffrion-Dyer-Feinberg algorithm. A formal presentation of this algorithm will be given in Chapter II. In anticipation of this, the specific class of problems to which the GDF algorithm applies and, in particular, to which this dissertation addresses itself will now be discussed.

1.5. The Problem to be Studied

The multiple objective optimization problem defined in Section 1.1 requires additional structure if it is to be resolved in the general framework of a man-machine interactive algorithm. To this end, the specific problem of interest is as follows:

$$\text{maximize} \quad U[f_1(\underline{x}), f_2(\underline{x}), \ldots, f_r(\underline{x})] \tag{1.3}$$

subject to

$$\underline{x} \in X = \{\underline{x} \mid g_j(\underline{x}) \leq 0, \quad j=1,2,\ldots,m\} \tag{1.4}$$

where $\underline{x}$ is an n-dimensional vector of decision variables, $f_i(\underline{x})$, $i=1,2,\ldots,r$, are r distinct objective functions of the decision vector $\underline{x}$, X forms the constrained set of feasible decisions, and U is the decision maker's overall preference function defined on the range of $\underline{f}$.

The set X and each f_i have been previously defined by the decision maker. The overall preference function U, however, is not identified explicitly; rather, it is established locally at various iteration points by questioning the decision maker.

It is assumed that the functions U, f_i, and g_j are continuously differentiable. The functions U and f_i are concave while the functions g_j are convex. Furthermore, U is increasing in each f_i; and X is a nonempty, compact, convex set.

Since U is increasing in $\underline{f}$, the optimal solution to (1.3)-(1.4) is an efficient point. Because the optimal solution also maximizes the decision maker's overall preference function, it is the best-compromise solution. Conversely, the best-compromise solution is obviously optimal. Hence, problem (1.3)-(1.4) constructively characterizes the best-compromise solution.

It is interesting to note that even under this characterization, it is impossible to disregard any efficient point. The following lemma by Soland shows that every efficient point is still a candidate for the best-compromise solution.

Lemma 1.1 [52]

For every efficient point $\underline{x}^e$ there exists an overall preference function U defined on R^r which is nondecreasing, concave, and possesses continuous derivatives of arbitrary order s, such that $U(\underline{f}(\underline{x}))$ is maximized over X at $\underline{x}^e$ and $U(\underline{f})$ is maximized over Y uniquely at $\underline{f}^e = \underline{f}(\underline{x}^e)$.

1.6. Additional Assumptions

The objective functions, f_i, should also satisfy two additional assumptions. The first of these states that all objectives are relevant to the decision maker's preferences. This simply means that each objective function is germane to the decision maker's judicial process of determining preferences among the points in the objective space. Stated mathematically,

$$\frac{\partial U}{\partial f_i} \not\equiv 0 \quad \text{for } i=1,2,\ldots,r.$$

In cases where irrelevant objectives exist, they should be omitted since they in no way affect the decision maker's preferences.

The second assumption requires the objectives to be functionally independent. This refers to the fact that there must not exist a functional relationship among any of the objectives used. Essentially, this precludes redundancy from occurring. More formally, for any set of objective functions, there is no non-zero function H of these objectives for which $H(\underline{x}) \equiv 0$ for all $\underline{x}$ in X.

To illustrate why functional independence is necessary, consider the following example. Suppose there are three objective functions f_1, f_2, and f_3. In addition, assume that $f_1 = 5f_2$. Then, the first two of these objectives are related by the function

$$H(f_1,f_2) = f_1 - 5f_2 = 0 .$$

Thus, they are functionally dependent.

Now suppose one would want to investigate the tradeoffs between f_1 and f_3, for example. Since f_2 is necessarily fixed at some level, f_1 must also be fixed. Thus, it becomes meaningless to establish tradeoffs between f_1 and f_3.

On the other hand, functional independence among f_1, f_2, and f_3 would make it reasonable to consider such a tradeoff. In fact, it would permit us to consider tradeoffs between <u>any</u> pair of these objectives.

1.7. Superiority Test

In attempting to solve the multiple objective optimization problem presented in Section 1.5, it would be advantageous to be able to repeatedly restrict the decision space to those points which represent favorable alternatives to the decision maker. In this way, the decision maker would avoid considering courses of action which are inferior to his present state. With this aim, Lemma 1.2 is presented. Although it is not as discriminatory as one might hope, it does reveal a very effective test. This lemma and its contrapositive are repeatedly referred to in later chapters.

Lemma 1.2

Suppose U is concave and differentiable on E^n. Then for any scalar $\alpha > 0$,

$$\alpha \nabla_{\underline{x}} U(\underline{x}^2)^t \cdot (\underline{x}^1 - \underline{x}^2) < 0 \quad \text{implies } U(\underline{x}^1) < U(\underline{x}^2).$$

Proof.

Since $\alpha \nabla_{\underline{x}} U(\underline{x}^2)^t \cdot (\underline{x}^1 - \underline{x}^2) < 0$ and $\alpha > 0$,

$$\nabla_{\underline{x}} U(\underline{x}^2)^t \cdot (\underline{x}^1 - \underline{x}^2) < 0. \tag{1.5}$$

U being concave and differentiable,

$$U(\underline{x}^1) \leq U(\underline{x}^2) + \nabla_{\underline{x}} U(\underline{x}^2)^t \cdot (\underline{x}^1 - \underline{x}^2)$$

$$\Rightarrow U(\underline{x}^1) - U(\underline{x}^2) \leq \nabla_{\underline{x}} U(\underline{x}^2)^t \cdot (\underline{x}^1 - \underline{x}^2).$$

Using (1.5), it follows that

$$U(\underline{x}^1) < U(\underline{x}^2).$$

Q.E.D.

By successfully passing this test (i.e., $\alpha \nabla_{\underline{x}} U(\underline{x}^2)^t \cdot (\underline{x}^1 - \underline{x}^2) < 0$), one has necessarily found a point $\underline{x}^2$ which is preferred to $\underline{x}^1$. Moreover, the function U does not have to be known explicitly in order to make this test; any positive multiple of its gradient is sufficient.

This lemma is also instrumental in defining a region which contains all points of promising value in terms of U. By taking the contrapositive of the above lemma, namely

$$U(\underline{x}) \geq U(\underline{x}^*) \Rightarrow \alpha \nabla_{\underline{x}} U(\underline{x}^*)^t \cdot (\underline{x} - \underline{x}^*) \geq 0,$$

we see that to find any point $\underline{x}$ which has a value equal to or greater than $U(\underline{x}^*)$, the search need only concentrate in that half-space for which

$$\alpha \nabla_{\underline{x}} U(\underline{x}^*)^t \cdot (\underline{x} - \underline{x}^*) \geq 0.$$

Again to emphasize, the function U need not be known explicitly. The gradient of U at $\underline{x}^*$ or a positive multiple of it is sufficient to determine this inequality.

1.8. A Usable Direction

From a given feasible solution in decision space, a direction will be determined which will align itself with preferred solutions. Exactly how this is accomplished is the subject of Chapters II and III. However, before this discussion can take place, the concept of a direction must be made more precise.

Definition. (Direction)

Any non-zero, n-dimensional vector $\underline{d}$ is a direction.

Definition. (Feasible Direction)

Let $\underline{x} \in X \subset E^n$, where X is a convex set. A direction $\underline{d}$ at $\underline{x}$ is feasible if and only if $\underline{x} + \lambda\underline{d} \in X$ for sufficiently small positive λ.

Definition. (Usable Direction)

A feasible direction which improves the overall objective function is said to be usable. In particular, $\underline{d}$ is a usable direction if it is a feasible direction which also satisfies

$$\nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{d} > 0,$$

where U is a differentiable concave function.

By repeatedly moving in a usable direction from one feasible point to another, it is hoped that one's location in decision space will continually improve and that the process will eventually terminate. Earlier, a test (Lemma 1.2) was presented which showed how one can determine if he has improved his position in decision space. Chapter III will explain in what way this relates to the multiple objective optimization problem. As for process termination, this is the subject of the next section.

1.9. Algorithmic Convergence

The discussion here concerning the concept of an algorithm and the meaning of its convergence is attributed to Zangwill [62]. He points out in his book, entitled Nonlinear Programming: A Unified Approach, that determining an optimal point, except in very special cases, is not a finite process. Unless one fortuitously finds an optimal point, the

iterative process of going from one feasible solution to the next will never end. For this reason it must be made clear what is meant by the expression "a convergent algorithm".

To begin, we define what is known as a point-to-set map.

Definition. (Point-to-Set Map)

A point-to-set map A_k: $V_k \rightarrow V_{k+1}$ is a map which associates with any point $z \in V_k$, a set $A_k(z)$ in V_{k+1}.

From this, it is possible to define the notion of an algorithm.

Definition. (Algorithm)

An algorithm is an iterative process consisting of a sequence of point-to-set maps

$$A_k : V_k \rightarrow V_{k+1}.$$

Given an initial point $z^1 \in V_1$, assume that the points $z^2,\ldots,z^k$ have been generated. The algorithm then proceeds as follows:

If $A_k(z^k) = \varphi$, the process stops.

Otherwise, select the next point in the sequence by use of the recursion

$$z^{k+1} \in A_k(z^k)$$

where z^{k+1} is any point in the set $A_k(z^k)$.

Under this definition any arbitrary sequence of points $y^1,y^2,\ldots,y^i,\ldots$ can be construed as an algorithm. Simply define $z^1 = y^1$, $V_i = \{y^i\}$, and $A_i(z^i) = \{y^{i+1}\}$ for all i. Clearly, this algorithm generates the desired sequence.

This general interpretation of an algorithm is useful in proving convergence. As Zangwill indicates, given a cumbersome or complicated algorithm which generates a sequence of points, it is often easier to redefine the sequence as a second algorithm. In establishing convergence for this second algorithm, one has necessarily implied convergence of the original algorithm.

Because algorithmic convergence generally occurs only in a limiting sense, it is prudent to be precise in discussing the limit of a sequence.

Definition. (Limit)

A sequence $z^1, z^2, z^3, \ldots$, has a limit $\underline{\ell}$ if for any $\epsilon > 0$, $\exists$ N such that $\forall$ k $\geq$ N

$$\| \underline{z}^k - \underline{\ell} \| < \epsilon,$$

where

$$\| \underline{z}^k - \underline{\ell} \| \equiv \left(\sum_{i=1}^{n} (z_i^k - \ell_i)^2 \right)^{1/2}.$$

Definition. (Convergent)

A sequence is convergent if it has a limit.

To say that a sequence has a limit $\underline{\ell}$ is to imply that all but a finite number of terms of that sequence are within ϵ of $\underline{\ell}$. A much weaker requirement is to state that infinitely many terms of the sequence are within ϵ of $\underline{\ell}$.

Definition. (Limit Point)

A sequence $\underline{z}^1, \underline{z}^2, \underline{z}^3, \ldots$, has a limit point (cluster point, accumulation point) $\underline{\ell}$ if for any $\epsilon > 0$ and any N, $\exists$ k $\geq$ N such that

$$\| \underline{z}^k - \underline{\ell} \| < \epsilon .$$

Although a limit is a limit point, the converse is not necessarily true. For example, define $z^k = (-1)^k$. The sequence $\underline{z}^1, \underline{z}^2, \underline{z}^3, \ldots$, has limit points +1 and -1, but it has no limit.

Lemma 1.3 further explains the relationship between limit points and limits.

Lemma 1.3 [49]

A sequence $\underline{z}^1, \underline{z}^2, \underline{z}^3, \ldots$, has a limit point $\underline{\ell}$ if and only if there exists a subsequence of this sequence which converges to $\underline{\ell}$.

The definition of a convergent algorithm can now be stated. Zangwill asserts that the following definition is reasonable, generally applicable, and very useful.

Definition. (Convergent Algorithm) [62]

Given a problem and a solution set $\Omega \subset E^n$, a convergent algorithm is an algorithm with the following properties:

(a) If the algorithm stops at a point $\underline{z}$, then it indicates either that no solution exists or that $\underline{z}$ is a solution. Also, if a point $\underline{z}$ is a solution, then either $A_k(\underline{z}) = \varphi$ or $\underline{y} \in A_k(\underline{z})$ implies $\underline{y}$ is a solution.

(b) Assume the algorithm generates an infinite sequence of points none of which is a solution. Then if all points are not on a compact set, no solution point exists, while if all

points are on a compact set, the limit of any convergent subsequence is a solution.

In essence, an algorithm converges if it either calculates the optimal point or is able to determine it in a limiting sense. On the other hand, if an algorithmic sequence converges to a non-optimal point, the algorithm is said not to converge. This definition of algorithmic convergence will be used in Chapter III to show that a particular algorithmic process will eventually converge.

1.10. Organization of the Report

As previously mentioned, Chapter II details the Geoffrion-Dyer-Feinberg Interactive algorithm. Man-machine interactive decisions are its salient feature. The algorithm is described in the context of the Frank-Wolfe mathematical programming algorithm. Some of the limitations of this algorithm are presented in order to emphasize the reasons why one should consider the alternative approach discussed in the next chapter.

The Tradeoff Cutting Plane algorithm is introduced in Chapter III. The intent of the algorithm and the concepts behind it are also explained. A detailed presentation of the algorithm and a proof of its convergence are included. The chapter concludes with a discussion about a sensitivity analysis which is automatically performed upon termination of the algorithm.

In order to illustrate an application of the Tradeoff Cutting Plane algorithm to a specific example, a storm drainage problem, similar to that faced by many medium size communities in the United States, is presented in Chapter IV. Convergence results are shown for various policies associated with this application.

The discrete multiple objective optimization problem is considered in Chapter V. Although conceptually the optimization approach is the same, this chapter shows how the algorithm is modified to address this situation. The nature of the problem and its peculiar qualities are also discussed.

In conclusion, Chapter VI outlines possible areas for further research. It discusses areas relevant to the algorithm itself and multiple objective optimization in general.

In Appendix A an assortment of definitions and lemmas relevant to this report are listed. Appendix B follows with a presentation of a programming technique concerned with locating the optimal solution to a nonlinear programming problem. The technique is a specific type of those generally referred to as methods of centers, a clever cross between penalty function methods and methods of feasible directions. Finally, Appendix C displays a complete listing of the program.

CHAPTER II

THE GEOFFRION-DYER-FEINBERG INTERACTIVE ALGORITHM

In Section 1.4 it was stated that there are basically two categories of multiple objective techniques. In the category labeled "progressive articulation of preferences", reference was made to a man-machine interactive process termed the Geoffrion-Dyer-Feinberg (GDF) algorithm. In this algorithm use is made of tradeoffs between the objectives to determine the best-compromise solution.

2.1. General Overview

The method is essentially a search procedure which questions the decision maker at each step in order to advance to "improved" feasible solutions. In the performance of this search, explicit knowledge of the overall preference function is not essential. Instead, by questionning the decision maker, local information regarding his preferences is received; and this, in turn, is sufficient to determine a direction in which to proceed to preferred solutions. The step size is found by once again interacting with the decision maker to find the point on this ray which is most preferred. Termination occurs when the optimal is reached or when the improvement between steps is less than some specified value.

2.2. Description

This section will briefly describe the algorithm as it is presented by Geoffrion, Dyer, and Feinberg [22]. The technique was originally proposed by Geoffrion [21] in a more general framework. He, together with Dyer and Feinberg, later formalized the basic technique and applied it to the operation of an academic department.

The description of the algorithm is presented with specific reference to the Frank-Wolfe algorithm [19]. The reasons behind the selection of this particular algorithm include its simplicity, convenience, and appropriate theoretical properties [22].

Recall from Section 1.5 that the multiple objective optimization problem of interest is of the following form:

$$\text{maximize } U(f_1(\underline{x}),\ldots,f_r(\underline{x})) \qquad (2.1)$$

$$\text{subject to } \underline{x} \in X,$$

where $f_1, f_2, \ldots, f_r$ are r functionally independent objective functions defined on the convex, compact, feasible decision space X and U is the decision maker's overall preference function defined on the objective space Y. Each f_i is concave and continuously differentiable on X, and the preference function U is concave increasing and continuously differentiable on Y. The function U is not assumed to be known explicitly, whereas each f_i and the set X are assumed given.

In order to solve (2.1) via the Frank-Wolfe algorithm, one proceeds as follows:

Step 0. Choose an initial feasible solution $\underline{x}^k \in X$.

Let k=1.

Step 1. Determine an optimal solution $\underline{z}^k$ of the direction finding problem

$$\begin{aligned} &\text{maximize } \nabla_{\underline{x}} U[f_1(\underline{x}^k), f_2(\underline{x}^k), \ldots, f_r(\underline{x}^k)] \cdot \underline{z} \\ &\text{subject to } \underline{z} \in X . \end{aligned} \tag{2.2}$$

Let $\underline{d}^k = \underline{z}^k - \underline{x}^k$.

Step 2. Determine an optimal solution t^k of the step-size problem

$$\begin{aligned} &\text{maximize } U[f_1(\underline{x}^k + t\underline{d}^k), \ldots, f_r(\underline{x}^k + t\underline{d}^k)] \\ &\text{subject to } 0 \leq t \leq 1 . \end{aligned} \tag{2.3}$$

If the solution is optimal, terminate. Otherwise, let

$$\begin{aligned} &\underline{x}^{k+1} = \underline{x}^k + t^k \underline{d}^k , \\ &k = k+1, \end{aligned} \tag{2.4}$$

and return to Step 1.

In view of the fact that U is not known, this straightforward application of the Frank-Wolfe algorithm is presumptuous. Fortunately, adjustments can be made which circumvent the problem areas.

The first difficulty arises in Step 1 where the gradient of U is required. By applying the chain rule, one finds

$$\nabla_{\underline{x}} U[f_1(\underline{x}^k), \ldots, f_r(\underline{x}^k)] = \sum_{i=1}^{r} (\partial U/\partial f_i)^k \nabla_{\underline{x}} f_i(\underline{x}^k) \tag{2.5}$$

where $(\partial U/\partial f_i)^k$ is the i^{th} partial derivative of U evaluated at $[f_1(\underline{x}^k),\ldots,f_r(\underline{x}^k)]$, and $\nabla_{\underline{x}} f_i(\underline{x}^k)$ is the gradient of f_i evaluated at $\underline{x}^k$. Recalling that the optimal solution $\underline{z}^k$ of (2.2) is not affected by multiplying the objective function by a positive scalar, one may divide the objective function by any positive quantity (e.g., $(\partial U/\partial f_i)^k$).

Definition. (Reference Criterion) [22]

The objective function on which the decision maker bases his tradeoffs is termed the reference criterion.

Assume that f_1 is the reference criterion. Now by dividing equation (2.5) by $(\partial U/\partial f_1)^k$ and appropriately modifying the objective function in equation (2.2), Step 1 becomes equivalent to

$$\text{maximize} \quad \sum_{i=1}^{r} w_i^k \nabla_{\underline{x}} f_i(\underline{x}^k)^t \cdot \underline{z} \qquad (2.6)$$

$$\text{subject to } \underline{z} \in X,$$

where $w_i^k = (\partial U/\partial f_i)^k / (\partial U/\partial f_1)^k$ is the tradeoff ratio between objectives f_i and f_1 at the current point.

Once these tradeoffs are obtained from the decision maker, problem (2.2) can be solved. There are several methods by which these tradeoffs can be approximated. An exposition of these methods will not be presented here. Instead, one is referred to [15, 22].

Having completed Step 1, Step 2 is solved by graphically representing

$$f_i(\underline{x}^k + t\underline{d}^k), \quad 0 \le t \le 1$$

for all i=1,2,...,r, or by giving the value of the vector $[f_1(\underline{x}^k + t\underline{d}^k),\ldots,f_r(\underline{x}^k + t\underline{d}^k)]$ for various values of t between 0 and 1.

From this selection the decision maker chooses his preferred decision point which, as a result, solves problem (2.3). Step 2 is then completed, and the decision maker has the option of either terminating the process or returning to Step 1.

2.3. The Interactive Frank-Wolfe Algorithm

To be more concise, the algorithm works interactively with the decision maker in the following way:

Step 0. The decision maker chooses an initial point $\underline{x}^1 \in X$. Let k=1.

Step 1. a) The decision maker assesses his tradeoff ratios w_i^k at the current base point.

b) Compute an optimal solution $\underline{z}^k$ to problem (2.6). Let $\underline{d}^k = \underline{z}^k - \underline{x}^k$.

Step 2. The decision maker subjectively determines an optimal solution t^k to problem (2.3). He then either decides to terminate or proceeds on to Step 1 after redefining as in Equation (2.4).

2.4. Significant Attributes

Worthy of note is that the decision maker interacts only with matters regarding the objective space. Since the dimension of the objective space is usually significantly less than the dimension of the decision space, this fact is of practical importance. Furthermore, working only in the objective space, the decision maker determines tradeoffs between the objectives without being ostensibly influenced by the decision variables.

Secondly, as the decision maker progresses from iteration to iteration, he accumulates, either consciously or unconsciously, more

and more information. Consequently, he becomes increasingly more aware of the implications regarding his tradeoffs and the nature of his problem.

Finally, note that the decision maker only considers relative preferences; exact specification of the overall preference function is needless and impractical. The information of significant importance is the tradeoffs made between pairs of objectives. The r - 1 tradeoffs between objective i (i=2,3,...,r) and objective 1 are the minimum required. Consideration of all possible pairwise tradeoffs is unnecessary, unless one is interested in a check on consistency.

2.5. Some Limitations of the GDF Algorithm

For each cycle of the GDF algorithm, there are two interactions with the decision maker. Different types of information are required at each of these interactions.

The first interaction asks for estimates of the decision maker's marginal rates of substitution between the objectives. The difficulty in determining these estimates is of obvious concern. Dyer [15] has investigated this area and has developed a tradeoff routine designed to arrive at these estimates through ordinal comparisons. Even with this improvement, there is still difficulty in determining these values and a lack of confidence in the solution based on these estimates [57].

The second interaction requires the decision maker to resolve a step-size problem. Geoffrion [21] originally proposed that this be done by plotting the values of all r objectives $f_i(\underline{x}^k + t\underline{d}^k)$, i=1,2,...,r, as a function of t. As t varies between 0 and 1, the decision maker

examines the superimposed plots and selects the most preferred value of t. Geoffrion acknowledges that as the number of objective functions increases so also does the task of determining this optimal step-size.

Dyer [15] modified this procedure by displaying the objective vector for equally spaced values of t between 0 and 1. Once the decision maker selects his preferred vector, he has determined his step-size, and the algorithm is ready for another iteration.

Although these methods don't insure locating the maximum as requested in Step 2 of the GDF algorithm, Hogan [28] argues that the maximum is not essential for convergence. As long as a "big enough" improvement is made on each iteration, suboptimization of the step-size is permissible.

Next, some mention should be made of the fact that the Frank-Wolfe algorithm is the optimization procedure used to implement the Geoffrion-Dyer-Feinberg interactive program. Although it is not the only technique able to be incorporated, it was chosen for its computational simplicity and impressive convergence properties [28, 61]. Yet, the Frank-Wolfe procedure is limited by the fact that it addresses a nonlinear objective function subject to linear constraints. As stated in Oppenheimer [43], nonlinear constraints require the use of more complex nonlinear methods. The extent to which this might adversely affect the solution process is not clear.

Montgomery and Bettencourt [42], in applying the GDF algorithm to a study involving the effect of crew training on the effectiveness of a main battle tank, had to augment the GDF algorithm to accommodate constraints of the form

$$\sum_{j=1}^{n} x_j^2 = b .$$

They used a code [2] based on a cyclic coordinate algorithm to deal with this specific type of quadratic constraint.

Another important limitation of the Geoffrion-Dyer-Feinberg algorithm concerns the likelihood of transferring misleading information to the decision maker. Notice that in Step 1 of the algorithm, the objective function is maximized over the entire decision space X. Under the assumptions laid forth in Section 1.5, regions of this decision space can be removed from further consideration based on the decision maker's tradeoffs. (This will be further explained in Chapter III.) Consequently, the solution $\underline{z}^k$ in Step 1 might possibly be a point outside the region of interest. This, in turn, causes the decision maker (in Step 2) to needlessly consider points located in an unfavorable region.

Although theoretically the Geoffrion-Dyer-Feinberg algorithm is able to be used in conjunction with standard branch and bound procedures to affect a discrete solution, the amount of information required of the decision maker makes it an impractical approach [16]. Unless an improved branching scheme is developed, the extension of this algorithm to a discrete mathematical programming context is injudicious.

A number of the limitations explained here motivated the development of the Tradeoff Cutting Plane algorithm, which is covered in the next chapter. In describing this algorithm, reference is made to a methods of centers programming technique presented in Appendix B. A modification of this method is used as a subprocedure in the Tradeoff Cutting Plane algorithm. Consequently, occasional reference is made to Appendix B for further clarification. Coverage of the discrete optimization problem will be deferred until Chapter V.

CHAPTER III

THE TRADEOFF CUTTING PLANE ALGORITHM

In this chapter the Tradeoff Cutting Plane (TCP) algorithm is introduced and shown to converge. Although the algorithm is termed a cutting plane algorithm, it does not conform to the established meaning of this type of an algorithm. With cutting plane algorithms, each added constraint "cuts" off a portion of the previous region leaving the feasible region intact but eliminating the present "optimal" solution. The TCP algorithm, on the other hand, will cut across the feasible region and will retain both the present solution and the best-compromise solution. It can be viewed as an internal cutting plane approach as opposed to the former external approach. The author is aware of the possible terminological inconsistency, but believes that this purposeful abuse in terms will serve to simplify the presentation, not cause any unwarranted confusion, and permit a more natural descriptive label of the technique.

First, the concepts behind the algorithm together with its intent are discussed.

3.1 Intent of the Tradeoff Cutting Plane Algorithm

As formerly mentioned, the TCP algorithm was motivated by a number of the problems associated with the GDF algorithm. Primary

concern centered on the problems of tradeoff sensitivity, step-size resolution, and discrete optimization. By addressing these problems, it was anticipated that the needs and judgmental capacities of the decision maker would be better served. This should not be construed as a condemnation of the GDF algorithm for the TCP algorithm is closely related to this approach. Besides, the GDF algorithm has been shown to be a useful and innovative tool [22, 43]. Instead, what is presented here should be viewed as a viable alternative to GDF method.

The information requirements of the two algorithms are different. Whereas the GDF algorithm requires interaction with the decision maker on each iteration to resolve the step-size problem and to acquire tradeoffs, the TCP algorithm only requires the tradeoff interaction. In other words, the TCP algorithm seeks to make more efficient use of the available information. By eliminating the need for a step-size interactive decision, one type of value judgment is completely removed. This is significant in view of the fact that the aim of man-machine interactive algorithms is to minimize the information burden of the decision maker while still allowing for an effective solution process.

As noted, the task of supplying tradeoffs between objectives is continued in the TCP algorithm. Its purpose here, however, is to establish a cutting plane in objective space in order to remove from further consideration large areas of the decision space. Although the decision maker will not be aware of the extent to which he has reduced his decision space, he is assured that the best-compromise solution has not been carelessly eliminated. This reduced space is then decreased again by another cut. By continuing

this process, the decision maker tends towards the area of the original decision space which contains best-compromise solution.

The tradeoffs supplied by the decision maker function to a large extent as a means of further restricting the decision space. The cutting plane established from these tradeoffs focuses in on the most relevant portion of the decision space, but the accomplishment of this is based more on eliminating the inferior portion of the space than on identifying the superior portion. Simply put, the cut acts as a restriction on where not to proceed. Upon termination of the algorithm, the decision maker has identified the neighborhood of the best-compromise solution, if not the solution itself.

This cutting plane approach also makes possible a post-optimal analysis of the decision maker's tradeoffs. Assessing value tradeoffs between objectives has always been a troublesome and difficult task. Although this problem of tradeoff assessment still remains, it is somewhat lightened by the fact that the tradeoffs are able to be tested against "critical" limits. It gives the decision maker an indication of the amount of latitude he had in making his tradeoffs.

An interesting by-product of the TCP algorithm is that the same solution procedure is used regardless of whether the original constraints are linear or not. The algorithm decomposes the evaluation process in a way similar to that of the GDF algorithm in that the overall optimization problem is divided into simpler subproblems. The original problem is actually restructured into a sequence of linear programs and one-dimensional searches. Each linear program requires the usual decision maker tradeoff interaction, whereas the

one-dimensional (step-size) searches require no decision maker interaction. As a result, the information burden on the part of the decision maker is reduced and a greater number of objectives are more easily accommodated.

3.2. Concepts Behind a Cut

At each iteration of the TCP algorithm, the decision maker evaluates one alternative which is represented by some point $\underline{f}' = \underline{f}(\underline{x}')$ in objective space. If one considers the r+1 dimensional space of the concave, overall preference function U and the objective space, the equation

$$U(\underline{f}) = U(\underline{f}') + \sum_{i=1}^{r} w_i(f_i - f_i') \tag{3.1}$$

where $w_i \equiv \partial U/\partial f_i \,|_{\underline{f}=\underline{f}'}$, is a linear approximation of U at $U(\underline{f}')$. It describes the r dimensional hyperplane which is tangent to the indifference surface at $\underline{f}'$. Focusing on just the r dimensional objective space, the r-1 dimensional hyperplane satisfying

$$\sum_{i=1}^{r} w_i(f_i - f_i') = 0 \tag{3.2}$$

provides a boundary between those values of U which are promising and those which are unfavorable. A promising region refers to a region containing at least those points whose values in terms of U is equal to or greater than $U(\underline{f}')$. Thus, in searching for improved values of U, one need only concentrate in that half-space for which

$$\sum_{i=1}^{r} w_i(f_i - f_i') \geq 0 . \tag{3.3}$$

This is what is discussed in Section 1.7 with regards to the contrapositive of Lemma 1.2. The difference here being that the half-space is in objective space and not, as before, in decision space.

With the $\partial U/\partial f_i$, for i=1,2,...,r, being unknown, the weights w_i and, therefore, inequality (3.3) cannot be directly determined. However, if one were to divide this inequality by the positive scalar $(\partial U/\partial f_1)\,|_{\underline{f}=\underline{f}'}$, it would read like this:

$$\sum_{i=1}^{r} [(\partial U/\partial f_i)/(\partial U/\partial f_1)](f_i - f_i') \geq 0. \qquad (3.4)$$

Since $(\partial U/\partial f_i)/(\partial U/\partial f_1)\,|_{\underline{f}=\underline{f}'}$, for i=2,3,...,r, are the decision maker's tradeoffs at $\underline{f}'$, inequality (3.4) is known. Furthermore, because those $\underline{f}$ which satisfy (3.3) also satisfy (3.4) and vice versa, the promising region of the objective space is now known.

It should be emphasized that need for exact knowledge of U's functional form has been circumvented. The tradeoffs furnished by the decision maker are sufficient information to further restrict the space containing the best-compromise solution.

In Section 1.2 it was stated that the slope of an indifference curve is negative. That is, while holding all other objectives constant, an indifference change in the values of objective 1 and i by the respective amounts Δ_1 and Δ_i forms the positive tradeoff ratio

$$-\Delta_1/\Delta_i = (\partial U/\partial f_i)/(\partial U/\partial f_1). \qquad (3.5)$$

The sum in (3.4) is a linear combination of concave functions. Since the weights $((\partial U/\partial f_i)/(\partial U/\partial f_1)$, $i=2,3,\ldots,r)$ of this linear combination are positive, the left-hand-side of the expression in (3.4) is itself a concave function (Appendix A, Lemma A.4). In requiring this expression to be greater than or equal to zero, the corresponding set of feasible decisions which satisfy this inequality is convex (Appendix A, Lemma A.2).

Succinctly, the process works as follows. The decision maker furnishes tradeoffs at each iteration. From these tradeoffs a cut is formed in objective space. This restriction, which limits the promising region, defines a convex set in decision space. The intersection of this set with the initial feasible region in the decision space defines a convex set which must contain the best-compromise solution.

To illustrate, let's assume that there are two objectives, f_1 and f_2, defined in two decision variables, $\underline{x}^1$ and $\underline{x}^2$. Figure 3.1 shows an original convex set, X, before any tradeoffs are made. Figure 3.2 shows the corresponding set, Y, in objective space. The current point is $\underline{x}'$ in X. This means that the decision maker furnishes tradeoffs at the point $\underline{f}(\underline{x}')$ in Y. From these tradeoffs the half-space of promising points is established in objective space. This half-space cut translates to the convex set diagrammed in Figure 3.3. The shaded region in Figure 3.2 illustrates the promising region; the one in Figure 3.3 shows the restricted region where the best-compromise solution is located.

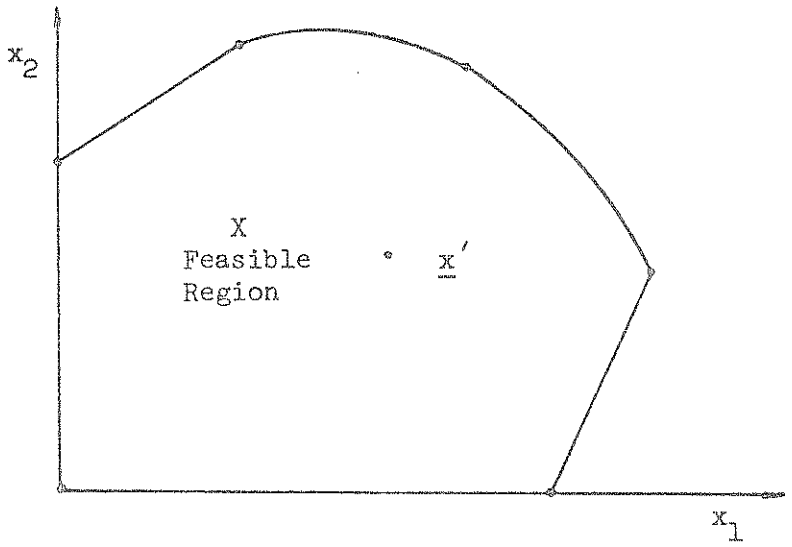


Figure 3.1 Decision Space

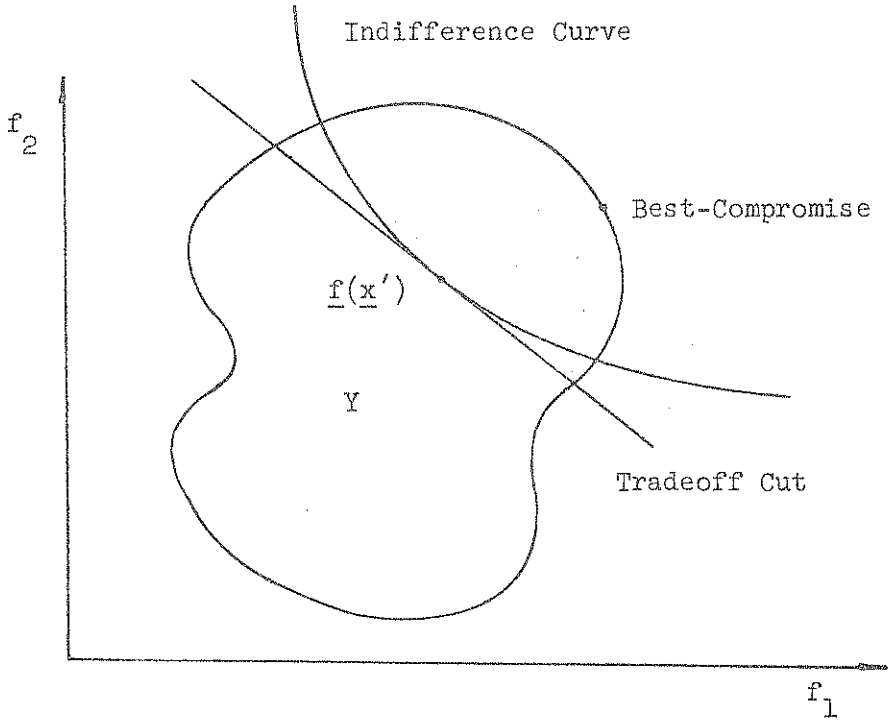


Figure 3.2 Objective Space

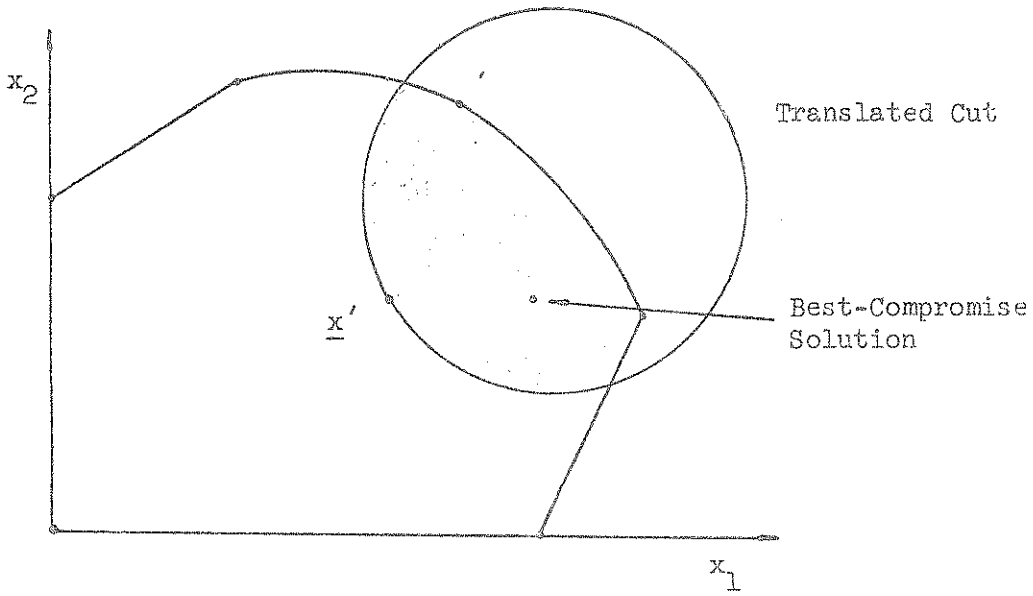


Figure 3.3 Translated Cut in Decision Space

3.3. Locating the Next Tradeoff Point

The question which now arises is how does one select a new point in the promising region? The problem relates to that of trying to locate a point in the restricted region of the decision space.

The point $\underline{x}'$ in X as well as the equation of the constraint through this point (introduced by the translated cut) are known. The constraints associated with the original feasible region are also known. A point near the "center" of this constrained set is desirable for it would eliminate an ample portion of this set on the next cut. A method of centers technique is applicable (Appendix B).

Before considering how this is accomplished, we should be assured that if one locates a point in this restricted set, then its image under the mapping $\underline{f}$ is in the promising region. Lemma A.7 in Appendix A is directed at this specific situation. It, in effect, states that no matter where one locates in the restricted set of the decision space, the corresponding point in objective space must be in the promising region.

Another pertinent question is whether or not the restricted set of the decision space contains the best-compromise solution. Lemma A.8 answers this question, for it states that if there is a best-compromise in objective space, then a solution exists, and it must be located within the restricted set of the decision space.

Now let's proceed in trying to locate this best-compromise solution. The image of this solution is obviously on the boundary of the feasible region in objective space. However, a decision

point whose image is on the boundary is not necessarily found on the border of the feasible set in decision space. Thus, to concentrate on the extreme points or even to focus on the boundaries of the decision space would be a mistake. One must consider the entire restricted region when searching for the best-compromise solution. Consequently, a method similar to the methods of centers technique discussed in Appendix B will be used.

Given that we are located at some point in the decision space and the constraint through this point has been determined, we move along a ray emanating from here until the location is reached at which the maximum distance from any constraint is minimized. The image of this new point is the next tradeoff point.

Definition. (Tradeoff Point)

A tradeoff point is any point in objective space at which the decision maker assesses his tradeoffs.

The direction of this ray is very important. It must orient itself in a feasible direction which will increase the unknown, overall preference function U. To select this usable direction $\underline{h}$ at a point $\underline{x}^\circ$, consider the following linear programming problem:

$$\text{minimize} \quad M \tag{3.6}$$

$$\text{subject to} \quad -M - \alpha \nabla_{\underline{x}} U(\underline{x}^\circ)^t \cdot \underline{h} \leq 0 \tag{3.7}$$

$$-M + g_i(\underline{x}^\circ) + \nabla_{\underline{x}} g_i(\underline{x}^\circ)^t \cdot \underline{h} \leq 0$$

$$i=1,2,\ldots,m_k \tag{3.8}$$

$$-1 \leq h_j \leq 1, \quad j=1,2,\ldots,n \tag{3.9}$$

where

$$\alpha \equiv 1/(\partial U/\partial f_1)\,\Big|_{\underline{f}=\underline{f}(\underline{x}^\circ)}.$$

Note the similarity with the LP problem (B.6) in Appendix B. The constraints (3.8) insure feasibility, whereas (3.7) guarantees that the vector $\underline{h}$ points in a direction which will increase U. Since U is unknown, constraint (3.7) is not exactly like the one found in (B.6). Instead, it reflects the known, local information the decision maker furnishes about U. Consequently, interaction with the decision maker is required to establish the LP problem (3.6)-(3.9).

For example, let there be three objective functions associated with the multiple objective problem:

$$f_1 = 2x_1 + x_2 ,$$

$$f_2 = -x_1^2 - x_2^2 + 10 ,$$

$$f_3 = 4x_1 + 6x_2 - 2x_1^2 - 2x_1x_2 - 2x_2^2 .$$

Assume that f_1 is the reference criterion. Now suppose the decision maker gives (1,1/2,5) as the tradeoff vector at the current point $\underline{f}(\underline{x}^o) = \underline{f}(1,2) = (4,5,2)$. With this information, constraint (3.7) becomes

$$-M - \alpha \nabla_{\underline{x}} U(x^o)^t \cdot \underline{h} \qquad (3.10)$$

$$= -M - (1,1/2,5) \cdot \begin{bmatrix} 2 & 1 \\ -2x_1 & -2x_2 \\ 4 - 4x_1 - 2x_2 & 6 - 2x_1 - 4x_2 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \qquad (3.11)$$

$$= -M - (1,1/2,5) \cdot \begin{bmatrix} 2 & 1 \\ -2 & -4 \\ -4 & -4 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \qquad (3.12)$$

$$= \quad -M - (-19, -21) \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \tag{3.13}$$

$$= \quad -M + 19h_1 + 21h_2 \leq 0 . \tag{3.14}$$

The other constraints are derived directly from the previously known information. The LP problem (3.6)-(3.9) can now be solved to determine the usable direction $\underline{h}$.

It should be noted that for any $\underline{x}$ in

$$C_k = \{\underline{x} \,|\, g_i(\underline{x}) \leq 0 , \quad i=1,2,\ldots,m_k\} ,$$

$(M,\underline{h}) = (0,\underline{0})$ is a feasible solution. Consequently, $M \leq 0$ for all $\underline{x}$ in C_k. Moreover, since U and g_i are continuously differentiable, M is the maximum of a set of continuous functions and, as such, is itself a continuous function. Use of these observations will be made later in proving convergence of the algorithm.

The constraints in (3.8) are associated with the original m constraints plus the tradeoff cuts which have been derived on each iteration. Assuming the current tradeoff point is $\underline{f}(\underline{x}^o)$, the left-hand-side of the last constraint in (3.8) (i.e., the one associated with the current tradeoff cut) is

$$-M + g_{m_k}(\underline{x}^o) + \nabla_{\underline{x}} g_{m_k}(\underline{x}^o)^t \cdot \underline{h}$$

$$= \quad -M - \sum_{i=1}^{r} t_i (f_i(\underline{x}^o) - f_i(\underline{x}^o))$$

$$- \quad \nabla_{\underline{x}=\underline{x}^o} \left[\sum_{i=1}^{r} t_i (f_i(\underline{x}) - f_i(\underline{x}^o)) \right]^t \cdot \underline{h}$$

$$\text{where } t_i \equiv \left(\frac{\partial U / \partial f_i}{\partial U / \partial f_1} \right) \Bigg|_{\underline{f} = \underline{f}(\underline{x}^o)}$$

$$= -M - \left[\sum_{i=1}^{r} t_i \nabla_{\underline{x}} f_i(\underline{x}^o) \right]^t \cdot \underline{h}$$

$$= -M - \left(\frac{1}{\partial U / \partial f_1} \right) \left[\sum_{i=1}^{r} (\partial U / \partial f_i) \nabla_{\underline{x}} f_i(\underline{x}^o) \right]^t \cdot \underline{h}$$

$$= -M - \alpha \nabla_{\underline{x}} U(\underline{x}^o)^t \cdot \underline{h} .$$

Thus, the last constraint in (3.8) is redundant with (3.7). In practice, (3.7) is actually dropped in favor of this newest constraint. This means that on each iteration of the algorithm the best feasible direction is determined directly from the "feasibility" constraints. The improvement of the decision maker's overall preference function U is given implicit consideration through one of these constraints.

3.4. Tradeoff Cutting Plane Algorithm

Having explained various phases of the Tradeoff Cutting Plane algorithm, the algorithm itself will now be presented. Let $m_0 = m$, the initial number of constraints associated with problem (1.3)-(1.4) in Section 1.5. Define

$$C_0 \equiv \{ \underline{x} \mid g_i(\underline{x}) \leq 0, \quad i=1,2,\ldots,m_0 \} .$$

At each iteration a new constraint is added, reflecting the tradeoff cut which was made in objective space. Therefore, after the k^{th} iteration, let

$$C_k \equiv \{ \underline{x} \mid g_i(\underline{x}) \leq 0, \quad i=1,2,\ldots,m_k \}$$

where $m_k = m_0 + k$ and $g_i(\underline{x}) \leq 0$, $i=1,2,\ldots,m_k$ represent the initial m constraints plus the k tradeoff cuts. The set C_k is assumed to have an interior and contain at least one point $\underline{x}^c$ for which $g_i(\underline{x}^c) < 0$, for all $i=1,2,\ldots,m_k$. This together with the fact that the functions $g_i(\underline{x})$ are convex and continuously differentiable ensures that the constraint qualification (Appendix A) holds at any feasible point $\underline{x}$ in C_k [27, 51, 62]. As before,

$$\alpha \equiv 1/(\partial U/\partial f_1)$$

evaluated at the current decision point. Let d be defined as the negative of the distance function (Appendix B)

$$d(\underline{y}) = \max \{ g_i(\underline{y}), \quad i=1,2,\ldots,m_k \}.$$

Two constants, ρ_1 and ρ_2, are used in the algorithm's step length determination schemes. They are set by the user prior to the algorithm's execution and are assumed to have values greater than one. The function of these constants will be explained after the algorithm has been stated.

Algorithm.

Step 0. Compute $\underline{x}^o \in C_o$ and set k=0.

Step 1. Determine tradeoffs at $\underline{f}(\underline{x}^o)$.

Step 2. Set $\underline{x} = \underline{x}^k$ and t=1.

Step 3. Solve (3.6)-(3.9) to obtain $(M(\underline{x}),\underline{h})$.

Step 4. If $M(\underline{x}) \geq 0$, STOP!

Otherwise, continue.

Step 5. Compute λ^*, the smallest positive scalar such that

$$d(\underline{x} + \lambda^* \underline{h}) = \min \{ d(\underline{x} + \lambda \underline{h}) \mid \lambda \geq 0 \} .$$

Step 6. Set $\underline{z} = \lambda^* \underline{h}$.

Step 7. If $\alpha \nabla_{\underline{x}} U(\underline{x} + t\underline{z})^t \cdot (-t\underline{z})$ $\begin{cases} < 0, \text{ go to Step 12} \\ = 0, \text{ go to Step 14} \\ > 0, \text{ continue.} \end{cases}$

Routine #1

Step 8. Set $t = t/\rho_1$.

Step 9. If $\alpha \nabla_{\underline{x}} U(\underline{x} + t\underline{z})^t \cdot (-t\underline{z}) \geq 0$, return to Step 8.

Otherwise, go to Step 14.

Routine #2

Step 10. If $\underline{x} + t\underline{z} \in C_k$, continue.

Otherwise, go to Step 13.

Step 11. If $\alpha \nabla_{\underline{x}} U(\underline{x} + t\underline{z})^t \cdot (-t\underline{z})$ $\begin{cases} < 0, \text{ continue} \\ = 0, \text{ go to Step 14} \\ > 0, \text{ go to Step 13.} \end{cases}$

Step 12. Set $t = \rho_2 t$ and return to Step 10.

Step 13. Set $t = t/\rho_2$.

Step 14. Set $\underline{x}^{k+1} = \underline{x} + t\underline{z}$, $k=k+1$, and return to Step 2.

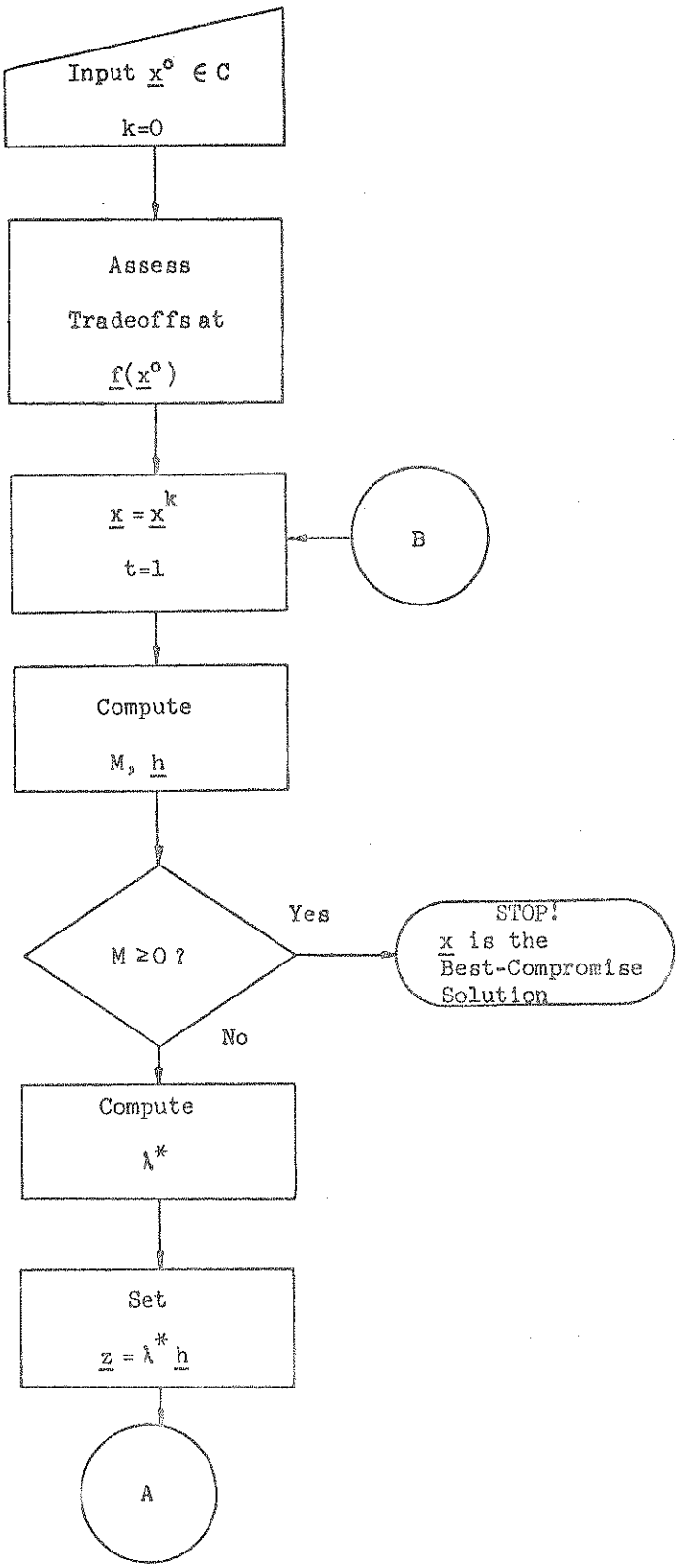


Figure 3.4a. Flowchart of the TCP Algorithm

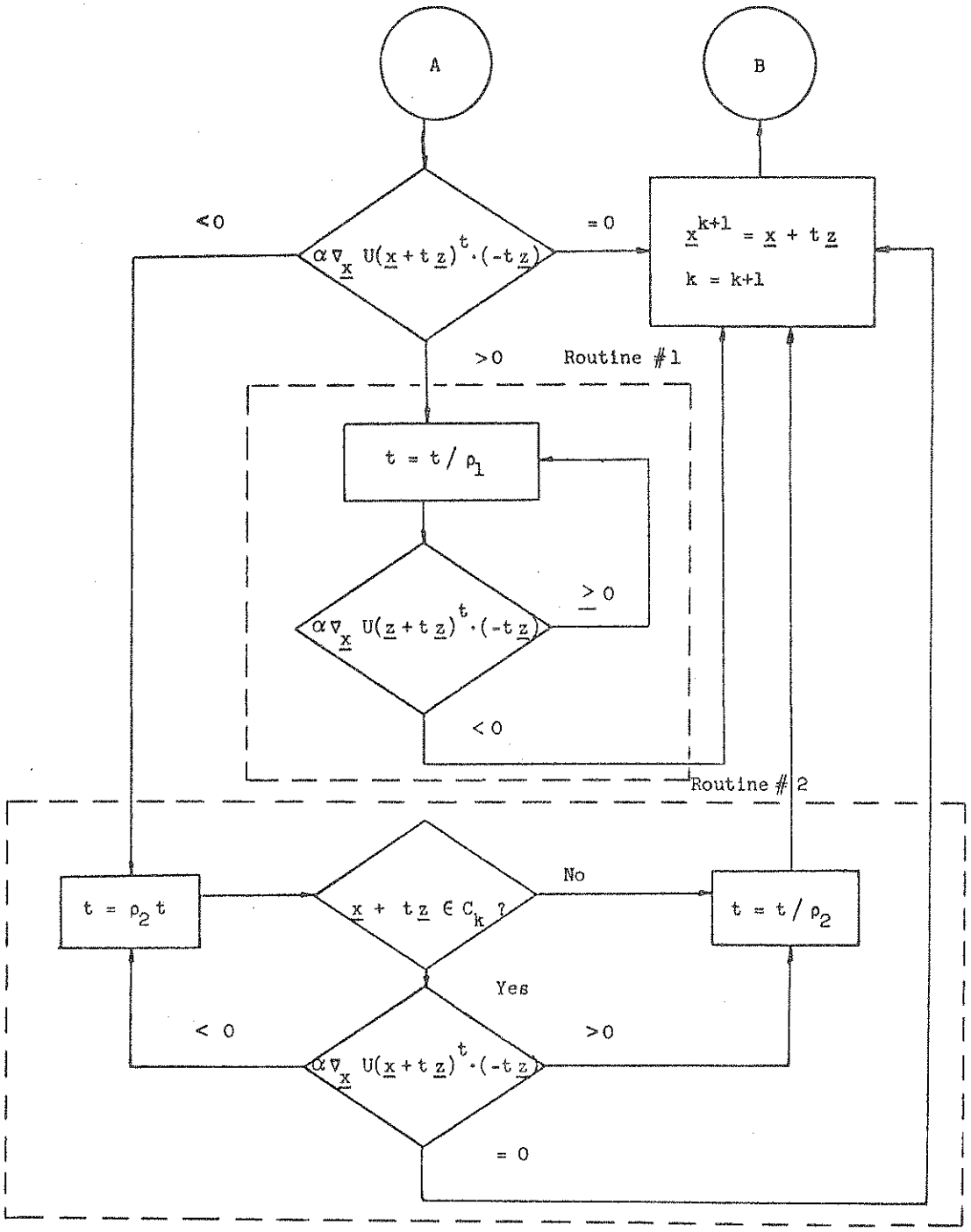


Figure 3.4b. Flowchart of the TCP Algorithm (Continued)

The TCP algorithm is composed of four parts. The first part, Steps 0-3, determines the termination index M and the usable direction $\underline{h}$ to be used in locating the center of the current feasible region. The next part is the algorithm's termination rule which is found at Step 4. The third part, Step 5, involves locating the center of the current feasible region. Finally, Steps 6-14 encompass the algorithm's step length determination schemes.

This last part is where the constants ρ_1 and ρ_2 appear. Specifically, they are used in the step length determination schemes denoted as routines #1 and #2. ρ_1 is seen in Step 8 of the algorithm. At this point in the process (Routine #1), too large a step-size has been taken. The result is that the point at which U reaches its maximum (along $\underline{x} + t\underline{z}$) has been passed. Since no statement can be made concerning the degree of improvement (if there has even been any), a decrease in the step-size is necessary. In Routine #1 the step-size is decreased by a factor of ρ_1 until the point at which U reaches its maximum has been passed again (this time in the other direction). The algorithm then sets the image of this new point as the next tradeoff point and continues.

In the present version of the algorithm, ρ_1 is set to 2. A higher value for ρ_1 would possibly result in reaching a point of guaranteed improvement in fewer iterations and, in turn, require fewer interactions with the decision maker. On the other hand, a smaller value for ρ_1 would allow for a "better" cut to be taken. By this it is meant that the cut intersects the path $\underline{x} + t\underline{z}$ at a point closer to where U reaches its maximum. In taking cuts close to this maximizing point, one reduces the feasible decision space more judiciously.

The other constant ρ_2 appears in Step 12. Entering Routine #2 of the TCP algorithm, the maximum of U along the path $\underline{x} + t\underline{z}$ has not been passed yet. Similar to the situation with ρ_1, the considerations one must address in determining a value for ρ_2 result in conflict. A large ρ_2 is beneficial, for the next step would most likely be infeasible and, consequently, not require any interaction on the part of the decision maker. Yet, a smaller value for ρ_2 would increase the possibility of obtaining a better cut.

As previously mentioned, Routines #1 and #2 of the algorithm are step length determination schemes. In both of these routines the final step length is less than or equal to the step length determined by the optimal gradient method (i.e., step length $\hat{\theta}$ where

$$U(\underline{x} + \hat{\theta} \nabla_{\underline{x}} U(\underline{x})) \equiv \max_{\theta} U(\underline{x} + \theta \nabla_{\underline{x}} U(\underline{x})).$$

This "undershooting" of the optimal gradient step is not as detrimental as it might at first appear. In fact, this tactic is often recommended with feasible direction methods to counteract the general tendency of this optimal step-size to overshoot the best step length in terms of the overall rate of convergence [51]. The best "relaxation" of this optimal step-size differs according to the objective function, but the fact remains that missing this optimal step-size on the short side usually yields improved overall rates of convergence. In practice, Simmons [51] has found that a relaxation factor of 0.9 furnishes generally good results. The relaxation factor associated with the TCP algorithm varies on each iteration. However, a shortened

step length is nearly always used. Exceptions to this occur when the optimal step-size corresponds with the center of the feasible region, in which case this step is retained.

In reference to the algorithm's flow chart, the bold box and diamonds indicate where interaction with the decision maker is required. Although these interactions appear at a number of places in the algorithm, one is precluded from interacting at all of these points due to the algorithm's logic. Note also that each interaction requests the same type of information, namely, local functional tradeoffs. These same tradeoffs are then used again in Step 3 to define the next LP problem.

Plainly, it is advantageous to limit the number of constraints necessary to solve for M in LP problem (3.6)-(3.9). The tradeoff cutting plane algorithm was designed with this in mind. As a result not all of the decision maker's tradeoffs are retained. Since each decision maker interaction produces tradeoffs which are in turn translated into constraints, it becomes clear that not necessarily all the information received from the decision maker continues to be pertinent.

For example, constraints derived from adjacent interaction points along the path $\underline{x} + t\underline{z}$ can produce nested subsets. The perceptive choice among the possible constraints to retain would be the final cut. This, in essence, is what is done in implementing the TCP algorithm.

The procedure works as follows. If the expression in Step 7 is zero, only one constraint (viz., the current constraint) is added. If this expression is <u>not</u> zero, one must enter either Routine #1 or Routine #2.

In Routine #1, the decision maker interactions continue until the same expression (viz., $\alpha \nabla U(\underline{x}+t\underline{z})^{t}\cdot(-t\underline{z})$) becomes non-positive. When this occurs, cuts produced by this last interaction and the previous one are retained. Stated differently, cuts are taken at both ends of an interval containing a maximizing point for U along the path $\underline{x}+t\underline{z}$.

In Routine #2, three different situations can occur. If a point is located for which the same test expression is zero, only this cut is retained. The second possibility occurs when the expression becomes positive before one experiences infeasibility. If this happens, the situation is the same as explained above for Routine #1. Finally, if infeasibility occurs (recall that the initial step is always feasible), the cut obtained on the last decision maker interaction is the only one retained.

Admittedly, the above scheme concerning what information to retain may not be the most prudent under all circumstances, but it does offer a sensible compromise between over specifying and inadequately defining the new feasible decision space.

3.5. Proof of Convergence

This section presents a convergency proof of the TCP algorithm. As explained in Section 1.9, convergence means that either the best-compromise point is located after a finite number of iterations or else it is determined in a limiting sense.

Prior to presenting the theorem on convergence, a number of lemmas are proven in order to facilitate a better understanding of the proof. In these lemmas reference is often made to t_{max}. Assuming U to be a concave function defined on a compact convex set C and $\underline{h}$ to be a usable direction at a point $\underline{x}$ in C, t_{max} determines a point

(viz., $\underline{x} + t_{max}\,\underline{h}$) at which

$$U(\underline{x} + t_{max}\,\underline{h}) = \max_{\substack{t \\ \underline{x} + t\underline{h} \in C}} U(\underline{x} + t\underline{h}) .$$

Two facts concerning t_{max} should be emphasized:

(i) t_{max} will always exist, and

(ii) since $\underline{h}$ is a usable direction, t_{max} will always be greater than zero.

Lemmas 3.1 and 3.2 give, in terms of the gradient of U, conditions of t for which the value of t is known in relationship to t_{max}.

Lemma 3.1

Let U be a differentiable concave function defined on the compact convex set C, and let $\underline{h}$ be a usable direction at $\underline{x} \in C$. If

$$\nabla_{\underline{x}} U(\underline{x} + t\underline{h})^{t} \cdot (-t\underline{h}) < 0 \quad \text{where } t \geq 0 ,$$

then $t \leq t_{max}$.

Proof.

Assume $t > t_{max}$ (>0). By definition, $U(\underline{x} + t\underline{h}) \leq U(\underline{x} + t_{max}\,\underline{h})$. As a result,

$$\nabla_{\underline{x}} U(\underline{x} + t\underline{h})^{t} \cdot [(\underline{x} + t_{max}\,\underline{h}) - (\underline{x} + t\underline{h})] \geq 0 \qquad \text{(Lemma A.6)}$$

$$\Rightarrow \nabla_{\underline{x}} U(\underline{x} + t\underline{h})^{t} \cdot [(t_{max} - t)\underline{h}] \geq 0$$

$$\Rightarrow \nabla_{\underline{x}} U(\underline{x} + t\underline{h})^{t} \cdot [-t\underline{h}] \geq 0 \qquad \text{since } t > 0 \text{ and } t_{max} < t .$$

But this contradicts the fact that

$$\nabla_{\underline{x}} U(\underline{x}+t\underline{h})^{t} \cdot (-t\underline{h}) < 0 .$$

Hence, $t \leq t_{max}$.

Q.E.D.

Lemma 3.2

Let U be a differentiable concave function defined on the compact convex set C, and let $\underline{h}$ be a usable direction at $\underline{x} \in C$. If

$$\nabla_{\underline{x}} U(\underline{x}+t\underline{h})^{t} \cdot (-t\underline{h}) > 0 \qquad \text{where } t \geq 0 ,$$

then $t_{max} \leq t$.

Proof.

Analogous to the proof of Lemma 3.1.

Q.E.D.

Lemmas 3.3 and 3.4 give statements concerning the value of U at $\underline{x}+t\underline{h}$, where t is contained in the closed interval $\left[\frac{t_{max}}{\rho}, t_{max}\right]$ and $\rho \geq 1$.

Lemma 3.3

Let U be a concave function defined on the compact convex set C, and let $\underline{h}$ be a usable direction at $\underline{x} \in C$. For any constant $\rho \geq 1$ and for all t such that $\frac{t_{max}}{\rho} \leq t \leq t_{max}$,

$$U(\underline{x}+t\underline{h}) \geq U(\underline{x} + \frac{t_{max}}{\rho}\ \underline{h}) .$$

Proof.

Since $\frac{t_{max}}{\rho} \leq t \leq t_{max}$, there must exist a λ such that $0 \leq \lambda \leq 1$ and

$$t = \frac{t_{max}}{\rho} + \lambda \left(t_{max} - \frac{t_{max}}{\rho}\right).$$

Thus,

$$U(\underline{x} + t\underline{h}) = U\left[\underline{x} + \left(\frac{t_{max}}{\rho} + \lambda\left[t_{max} - \frac{t_{max}}{\rho}\right]\right)\underline{h}\right]$$

$$= U\left[(1-\lambda)\left(\underline{x} + \frac{t_{max}}{\rho}\,\underline{h}\right) + \lambda\left(\underline{x} + t_{max}\,\underline{h}\right)\right]$$

$$\geq (1-\lambda)\, U\left(\underline{x} + \frac{t_{max}}{\rho}\,\underline{h}\right) + \lambda\, U\left(\underline{x} + t_{max}\,\underline{h}\right)$$

$$= U\left(\underline{x} + \frac{t_{max}}{\rho}\,\underline{h}\right) + \lambda\left[U\left(\underline{x} + t_{max}\,\underline{h}\right) - U\left(\underline{x} + \frac{t_{max}}{\rho}\,\underline{h}\right)\right].$$

Finally, since $\lambda \geq 0$ and $U(\underline{x} + t_{max}\,\underline{h}) - U(\underline{x} + \frac{t_{max}}{\rho}\,\underline{h}) \geq 0$,

$$U(\underline{x} + t\underline{h}) \geq U\left(\underline{x} + \frac{t_{max}}{\rho}\,\underline{h}\right).$$

Q.E.D.

Lemma 3.4

Let U be a concave function defined on the compact convex set C, and let $\underline{h}$ be a usable direction at $\underline{x} \in C$. For any constant $\rho \geq 1$ and for all t such that $\frac{t_{max}}{\rho} \leq t \leq t_{max}$,

$$U(\underline{x} + t\underline{h}) - U(\underline{x}) \geq \frac{1}{\rho}\left(\max_{\substack{t \\ \underline{x} + t\underline{h} \in C}} U(\underline{x} + t\underline{h}) - U(\underline{x})\right).$$

Proof.

Because $\frac{t_{max}}{\rho} \leq t \leq t_{max}$ and U is concave,

$$U(\underline{x}+t\underline{h}) \geq U(\underline{x} + \frac{1}{\rho}\, t_{max}\, \underline{h}) \qquad \text{(Lemma 3.3)}$$

$$= U[(1 - \frac{1}{\rho})\underline{x} + (\frac{1}{\rho})(\underline{x} + t_{max}\, \underline{h})]$$

$$\geq (1 - \frac{1}{\rho})\, U(\underline{x}) + \frac{1}{\rho}\, U(\underline{x}+t_{max}\, \underline{h})$$

$$= U(\underline{x}) + \frac{1}{\rho}\, [U(\underline{x}+t_{max}\, \underline{h}) - U(\underline{x})] .$$

Therefore, using the definition of t_{max},

$$U(\underline{x}+t\underline{h}) - U(\underline{x}) \geq \frac{1}{\rho} \left(\max_{\substack{t \\ \underline{x}+t\underline{h} \in C}} U(\underline{x}+t\underline{h}) - U(\underline{x}) \right) .$$

Q.E.D.

The next lemma refers to the step-size routine in the TCP algorithm and shows what improvement in U is guaranteed by this routine.

Lemma 3.5

Let U be a differentiable concave function defined on the compact convex set C. Let $\underline{x}' = \underline{x} + \theta'\underline{h}$ where $\underline{h}$ is a usable direction at $\underline{x} \in C$. Assume $\underline{h}$ solves the LP problem (3.6)-(3.9), and assume θ' is determined by Steps 7-13 of the TCP algorithm (i.e., $\underline{x}' \in A(\underline{x})$). Then

$$U(\underline{x}') - U(\underline{x}) \geq \frac{1}{\rho} \left[\max_{\substack{\varphi \\ \underline{x}+\varphi\underline{h} \in C}} U(\underline{x}+\varphi\, \underline{h}) - U(\underline{x}) \right]$$

where $\rho \equiv \max \{ \rho_1, \rho_2 \}$.

Proof. (Note: $\theta \underline{h} \equiv t \lambda^* \underline{h} \equiv t\underline{z}$)

Determining $\theta' \equiv t^\circ \lambda^*$ from Steps 7-13 of the TCP algorithm falls into three cases. The separate cases arise at Step 7 where $\alpha \nabla_{\underline{x}} U(\underline{x} + t\underline{z})(-t\underline{z})$ is tested to see if it is positive, negative, or zero.

Case 1: $\alpha \nabla_{\underline{x}} U(\underline{x} + t\underline{z})^t \cdot (-t\underline{z}) > 0$

By construction, $\underline{x} + t\underline{z}$ is initially in C_k. That is, the first step is always feasible. After this, t is repeatedly divided by ρ_1 until $\alpha \nabla_{\underline{x}} U(\underline{x} + t_{n+1}\,\underline{z})^t \cdot (-t_{n+1}\,\underline{z}) < 0$

where $t_m \equiv \dfrac{t}{\rho_1^m}$ for $m \geq 0$. Thus, by Lemma 3.1, $t_{n+1} \leq t_{max}$.

By construction, $\alpha \nabla_{\underline{x}} U(\underline{x} + t_n\,\underline{z})^t \cdot (-t_n\,\underline{z}) > 0$. Therefore, by Lemma 3.2, $t_{max} \leq t_n$. Consequently,

$$t_{max} \leq t / \rho_1^n$$

$$\Rightarrow \frac{t_{max}}{\rho_1} \leq \frac{t}{\rho_1^{n+1}} \qquad \text{(since } \rho_1 > 1\text{)}$$

$$\Rightarrow \frac{t_{max}}{\rho_1} \leq t_{n+1}.$$

As a result,

$$\frac{t_{max}}{\rho_1} \leq t_{n+1} \equiv t^\circ \leq t_{max}.$$

Finally, by Lemma 3.4,

$$U(\underline{x}+\theta'\underline{h}) - U(\underline{x}) = U(\underline{x}+t^{\circ}\Lambda^{*}\underline{h}) - U(\underline{x})$$

$$= U(\underline{x}+t^{\circ}\underline{z}) - U(\underline{x})$$

$$\geq \frac{1}{\rho_1}\left[\max_{\substack{t \\ \underline{x}+t\underline{z}\in C}} U(\underline{x}+t\underline{z}) - U(\underline{x})\right]$$

$$= \frac{1}{\rho_1}\left[\max_{\substack{\varphi \\ \underline{x}+\varphi\underline{h}\in C}} U(\underline{x}+\varphi\underline{h}) - U(\underline{x})\right]$$

$$\geq \frac{1}{\rho}\left[\max_{\substack{\varphi \\ \underline{x}+\varphi\underline{h}\in C}} U(\underline{x}+\varphi\underline{h}) - U(\underline{x})\right].$$

Case 2: $\alpha\, \nabla_{\underline{x}}\, U(\underline{x}+t\underline{z})^{t}\cdot(-t\underline{z}) < 0$

Again, $\underline{x}+t\underline{z}$ is initially in C_k. After this, t is continually increased by a factor of ρ_2 until either $\underline{x}+t_{n+1}\,\underline{z} \notin C_k$ or $\alpha\, \nabla_{\underline{x}}\, U(\underline{x}+t_{n+1}\,\underline{z})(-t_{n+1}\,z) \geq 0$ where $t_m \equiv \rho_2^m\, t$ for $m \geq 0$.

Subcase 2A: $\underline{x} + t_{n+1}\,\underline{z} \notin C_k$.

By construction, $\underline{x}+t_n\,\underline{z} \in C_k$ and $\alpha\, \nabla_{\underline{x}}\, U(\underline{x}+t_n\,\underline{z})^{t}\cdot(-t_n\,\underline{z}) < 0$. Thus, by Lemma 3.1, $t_n \leq t_{max}$.

Recall from the discussion concerning tradeoff cuts that for all $y \in C \cap \overline{C}_k$, $U(\underline{y}) < U(\underline{x})$. Therefore, along the path $\underline{x}+t\underline{z}$, U must achieve its maximum in C_k. Since $\underline{x}+t_{n+1}\,\underline{z} \notin C_k$, $t_{max} \leq t_{n+1}$.

Now, $t_{n+1} \geq t_{max}$

$$\Rightarrow \quad \rho_2^{(n+1)}\, t \geq t_{max}$$

$$\Rightarrow \quad \rho_2 (\, \rho_2^{n}\, t) \geq t_{max}$$

$$\Rightarrow \quad \rho_2^{n}\, t \geq \frac{t_{max}}{\rho_2} \qquad \text{since } \rho_2 > 1$$

$$\Rightarrow \quad t_n \geq \frac{t_{max}}{\rho_2} \; .$$

As a result,

$$\frac{t_{max}}{\rho_2} \leq t_n \equiv t^{\circ} \leq t_{max} \; .$$

Finally, by Lemma 3.4 and reasoning analogous to that found in Case 1,

$$U(\underline{x} + \theta' \underline{h}) - U(\underline{x}) \geq \frac{1}{\rho} \Big[\max_{\substack{\varphi \\ \underline{x} + \varphi \underline{h} \in C}} U(\underline{x} + \varphi \underline{h}) - U(\underline{x}) \Big] .$$

Subcase 2B: $\underline{x} + t_{n+1}\, \underline{z} \in C_k$ and $\alpha\, \nabla_{\underline{x}}\, U(\underline{x} + t_{n+1}\, \underline{z})^{t} \cdot (-t_{n+1}\, \underline{z}) > 0$. By construction, $\underline{x} + t_n\, \underline{z} \in C_k$ and $\alpha\, \nabla_{\underline{x}}\, U(\underline{x} + t_n\, \underline{z})^{t} \cdot (-t_n\, \underline{z}) < 0$. Hence, $t_n \leq t_{max}$ by Lemma 3.1.

Since $\alpha\, \nabla_{\underline{x}}\, U(\underline{x} + t_{n+1}\, \underline{z})^{t} \cdot (-t_{n+1}\, \underline{z}) > 0$, $t_{max} \leq t_{n+1}$ from Lemma 3.2.

Thus, $t_n \leq t_{max} \leq t_{n+1}$. Therefore, as in Subcase 2A with $t_n \equiv t^{\circ}$,

$$U(\underline{x}+\theta'\underline{h}) - U(\underline{x}) \geq \frac{1}{\rho}\left[\max_{\substack{\varphi \\ \underline{x}+\varphi\underline{h}\in C}} U(\underline{x}+\varphi\underline{h}) - U(\underline{x})\right].$$

Subcase 2C: $\underline{x}+t_{n+1}\underline{z} \in C_k$ and $\alpha\, \nabla_{\underline{x}} U(\underline{x}+t_{n+1}\underline{z})^t\cdot(-t_{n+1}\underline{z}) = 0$.

Define $t_{n+1} \equiv t^\circ$ and refer to Case 3.

Case 3: $\alpha\, \nabla_{\underline{x}} U(\underline{x}+t\underline{z})^t\cdot(-t\underline{z}) = 0$

Define $t^\circ = t$. Then since $\alpha > 0$,

$$\nabla_{\underline{x}} U(\underline{x}+t^\circ\underline{z})^t\cdot(-t^\circ\underline{z}) = 0.$$

Furthermore, since $t^\circ > 0$,

$$\nabla_{\underline{x}} U(\underline{x}+t^\circ\underline{z})^t\cdot(\underline{z}) = 0.$$

Therefore, $\nabla_{\underline{x}} U(\underline{x}+t^\circ\underline{z})^t\cdot((t_{max}-t^\circ)\,\underline{z}) = 0$

$\Rightarrow \nabla_{\underline{x}} U(\underline{x}+t^\circ\underline{z})^t\cdot[(\underline{x}+t_{max}\,\underline{z}) - (\underline{x}+t^\circ\underline{z})] = 0$

$\Rightarrow U(\underline{x}+t_{max}\,\underline{z}) \leq U(\underline{x}+t^\circ\underline{z})$ by Lemma A.6.

By the definition of t_{max},

$$U(\underline{x}+t_{max}\,\underline{z}) \geq U(\underline{x}+t^\circ\underline{z}).$$

Consequently,

$$U(\underline{x}+t_{max}\,\underline{z}) = U(\underline{x}+t^\circ\underline{z}).$$

Hence,

$$U(\underline{x}') - U(\underline{x}) = U(\underline{x} + \theta' h) - U(\underline{x})$$

$$= U(\underline{x} + t^{\circ}\underline{z}) - U(\underline{x})$$

$$= \max_{\substack{t \\ \underline{x} + t\underline{z} \in C}} U(\underline{x} + t\underline{z}) - U(\underline{x})$$

$$\geq \frac{1}{\rho} \left[\max_{\substack{\varphi \\ \underline{x} + \varphi \underline{h} \in C}} U(\underline{x} + \varphi \underline{h}) - U(\underline{x}) \right].$$

These three cases resolve all possibilities. It follows, therefore, that the lemma's conclusion is valid.

Q.E.D.

In showing convergence of the TCP algorithm, use is made of a surrogate algorithm developed by Polak [46]. After comparing the TCP algorithm to this surrogate algorithm, the convergence theorem associated with this algorithm is employed to prove the convergence of the TCP algorithm. This approach makes possible the establishment of convergence without becoming intricately involved with the specific complexities of the TCP algorithm.

In his book Polak refers to points which satisfy some predetermined optimality condition as "desirable" points. His algorithm for locating these points in a closed subset T of $\mathbb{R}^n$ utilizes a set-valued search function A which maps T into the set of all non-empty subsets of T (i.e., $A: T \to 2^T$) and a stopping rule $f: T \to \mathbb{R}$.

Algorithm. (Polak) [46]

Step 0. Compute $\underline{x}^0 \in T$.

Step 1. Set $k = 0$.

Step 2. Compute $\underline{y} \in A(\underline{x}^k)$.

Step 3. Set $\underline{x}^{k+1} = \underline{y}$.

Step 4. If $f(\underline{x}^{k+1}) \geq f(\underline{x}^k)$ (or a direct test for determining $\underline{x}^k$ to be desirable), then set $\underline{x}^{k+1} = \underline{x}^k$ and stop. Otherwise, set $k = k+1$ and return to Step 2.

What follows is a convergence theorem which addresses this algorithm.

Theorem 3.1 [46]

Suppose that

(i) f is either continuous at all nondesirable points $\underline{x} \in T$, or else f is bounded below for $\underline{x} \in T$;

(ii) for every $\underline{x} \in T$ which is not desirable, $\exists\ \epsilon(\underline{x}) > 0$ and a $\delta(\underline{x}) < 0$ such that

$$f(\underline{x}'') - f(\underline{x}') \leq \delta(\underline{x}) < 0,$$

for all $\underline{x}' \in T$ such that $\| \underline{x}' - \underline{x} \| \leq \epsilon(\underline{x})$, and for all $x'' \in A(\underline{x}')$.

Then, either the sequence $\underline{x}^0, x^1, \underline{x}^2, \ldots$, constructed by Polak's algorithm is finite and its next to last element is desirable, or else it is infinite and every limit point of the sequence is desirable.

The next theorem makes use of the above result to show that the TCP algorithm converges to a desirable point. The proof follows, in part, the reasoning used in the modified method of centers proof found in [46].

Theorem 3.2

Let $\underline{x}^0, \underline{x}^1, \ldots, \underline{x}^k, \ldots$, be a sequence generated by the TCP algorithm. Then this sequence is either finite, ending with $\underline{x}^{k+1} = \underline{x}^k$ and $M(\underline{x}^k) \geq 0$, or it is infinite and every limit point $\underline{x}^*$ of this sequence satisfies $M(\underline{x}^*) \geq 0$.

Proof

Since the TCP algorithm only stops after a finite number of iterations if $M(\underline{x}^k) = 0$, the case of a finite sequence is straightforward.

Next, consider the case in which this sequence if infinite. In relationship to Theorem 3.1 (and Polak's algorithm), $T = C$ and $f = -U$. A is defined by the instructions in the TCP algorithm, and $\underline{x} \in C$ is termed desirable if $M(\underline{x}) \geq 0$. Theorem 3.1 will be shown to be satisfied by the maps f and A. As a result, every limit point $\underline{x}^*$ of this sequence will satisfy $M(\underline{x}^*) \geq 0$.

By assumption, U is continuously differentiable. Therefore, -U satisfies part (i) of Theorem 3.1.

To establish part (ii), it is necessary to show that for any $\underline{x}^* \in C$ such that $M(\underline{x}^*) < 0$, there exist an $\epsilon^* > 0$ and a $\delta^* < 0$ such that for all $\underline{x} \in D(\underline{x}^*, \epsilon^*) \equiv \{\underline{x} \in C \mid \|\underline{x}-\underline{x}^*\| \leq \epsilon^*\}$,

$$-U(\underline{x}') + U(\underline{x}) \leq \delta^*$$

for all $\underline{x}' \in A(\underline{x})$.

Recall that for any $\underline{x} \in C$, $M(\underline{x}) \leq 0$. Let $\underline{x}^*$ be any element in C for which $M(\underline{x}^*) \equiv \mu < 0$ (i.e., nondesirable). Since M is continuous, there exists an $\epsilon > 0$ such that

$$M(\underline{x}) \leq \mu/2 \quad \text{for all } \underline{x} \in D(\underline{x}^*, \epsilon). \tag{1}$$

By assumption, $\frac{\partial U}{\partial f_1} > 0$ for all $\underline{x} \in C$. U being continuously differentiable and C being compact, it follows from Lemma A.9 that

$$\gamma \equiv \inf \{ \frac{\partial U}{\partial f_1} \mid \underline{x} \in C \} > 0 . \tag{2}$$

Since $S \equiv \{ \underline{h} \in \mathbb{R}^n \mid |h_i| \leq 1, \quad i=1,2,\ldots,n \}$ is a compact set and U is continuously differentiable, there exists (Lemma A.10) an $\epsilon^* \in (0, \epsilon]$ and a $\lambda' > 0$ such that

$$| \nabla_{\underline{x}} U(\underline{x} + \lambda \underline{h})^t \cdot \underline{h} - \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} | \leq \frac{-\mu \gamma}{4} \tag{3}$$

for all $\underline{x} \in D(\underline{x}^*, \epsilon^*)$, for all $\underline{h} \in S$, for all $\lambda \in [0, \lambda']$.

Next, let $\underline{x} \in D(\underline{x}^*, \epsilon^*)$ be arbitrary. For this $\underline{x}$, let $\underline{h}$ be any vector in S which solves the linear programming problem (3.6)-(3.9). As a result of this and (1),

$$-\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} \leq M(\underline{x}) \leq \mu/2 . \tag{4}$$

Since $\alpha = 1/(\partial U / \partial f_1)$,

$$-\nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} \leq \mu/2\alpha \leq \mu\gamma/2 \tag{5}$$

By the Mean-Value Theorem (Theorem A.1), there exists an $\xi \in [0, \lambda']$ such that

$$U(\underline{x}+\lambda'\underline{h}) - U(\underline{x}) = \lambda' (\nabla_{\underline{x}} U(\underline{x}+\xi\underline{h})^t \cdot \underline{h}) . \qquad (6)$$

Making use of (6), (5), and (3)

$$\begin{aligned} - U(\underline{x}+\lambda'\underline{h}) + U(\underline{x}) &= \lambda'(- \nabla_{\underline{x}} U(\underline{x}+ \xi\underline{h})^t \cdot \underline{h}) \\ &= \lambda'[- \nabla_{\underline{x}} U(\underline{x}+ \xi\underline{h})^t \cdot \underline{h} - (-\nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h}) \\ &\qquad + (- \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h})] \\ &\le \lambda'[\left| - \nabla_{\underline{x}} U(\underline{x}+ \xi\underline{h})^t \cdot \underline{h} - (- \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h}) \right| \\ &\qquad + (- \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h})] \\ &\le \lambda' (- \frac{\mu\gamma}{4} + \frac{\mu\gamma}{2}) \\ &= \frac{\lambda'\gamma\mu}{4} < 0 . \end{aligned} \qquad (7)$$

Next let $\underline{x}' = \underline{x} + \theta'\underline{h}$ where θ' is determined by Steps 7-13 of the TCP algorithm (i.e., $\underline{x}' \in A(\underline{x})$). Because of Lemma 3.5 and (7),

$$\begin{aligned} - U(\underline{x}') + U(\underline{x}) &= - U(\underline{x} + \theta'\underline{h}) + U(\underline{x}) \\ &\le - \frac{1}{\rho} [\max_{\substack{\varphi \\ \underline{x}+\varphi\underline{h} \in C}} U(\underline{x}+\varphi\underline{h}) - U(\underline{x})] \\ &\le - \frac{1}{\rho} [U(\underline{x}+\lambda'\underline{h}) - U(\underline{x})] \quad \text{where } \lambda' \text{ is as in (6)} \\ &\le \frac{1}{\rho} (\frac{\lambda'\gamma\mu}{4}) \equiv \delta^* < 0 . \end{aligned} \qquad (8)$$

Since (8) holds for all $\underline{x} \in D(\underline{x}^*, \epsilon^*)$ and for all $\underline{x}' \in A(\underline{x})$, the theorem must be true.

Q.E.D.

Theorem 3.2 has just shown that the TCP algorithm converges. This raises the following question: to what type of point has the algorithm converged?

The optimality condition in the TCP algorithm concerns M. A sequence is constructed which is either finite and the last point $\underline{x}$ satisfies $M \geq 0$, or else it is infinite and has the property that any of its limit points must also satisfy this condition. The reason for this test is that

$$M(\underline{x}) \geq 0$$

$$\Updownarrow \qquad \text{(LP formulation)}$$

$$\min_{\underline{h} \in S} \left(\max\left(- \alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h}\,,\ g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h}\,,\ i=1,2,\ldots,m_k \right)\right) \geq 0$$

$$\Updownarrow \qquad \text{(Lemma A.11)}$$

$$\min_{\underline{h} \in S} \max_{i \in A(\underline{x})} \left\{ - \alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h}\,,\ \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} \right\} \geq 0$$

where $A(\underline{x}) \equiv \{ i \mid g_i(\underline{x}) = 0,\ i=1,2,\ldots,m_k \}$

$$\Updownarrow \qquad \text{(constraint qualification)}$$

$$- \alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} \geq 0 \qquad \text{for all } \underline{h} \in D(\underline{x})$$

where $D(\underline{x}) \equiv$ set of all feasible directions at $\underline{x}$

$$\Updownarrow \qquad (-\alpha < 0)$$

$$\nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} \leq 0 \qquad \text{for all } \underline{h} \in D(\underline{x})$$

$$\Updownarrow \qquad [62]$$

$\underline{x}$ is optimal to (1.3)-(1.4)

$$\Updownarrow \qquad \text{(Section 1.5)}$$

$\underline{x}$ is the best-compromise solution.

Thus, as anticipated, the TCP algorithm converges to the best-compromise solution.

3.6. Tradeoff Sensitivity

Decision makers are often unable to make confident assessments of their tradeoffs between objectives. They feel compelled (and rightly so) to justify their tradeoffs, making it at times extremely difficult for them to supply a specific value. Relaxing this requirement by asking instead a range within which the tradeoff should exist would be a more manageable request. On the other hand, this type of tradeoff ambiguity leads to an ill-defined problem. The decision maker's overall preference function becomes obscured, making it impossible to arrive at the best-compromise solution.

In connection with the TCP algorithm, a specific tradeoff value is necessary in determining the current tradeoff cut. Since past cuts are retained, a poor approximation could preclude one from ever finding the "true" best-compromise solution. A practical approach to avoid this type of problem would be to retain a given number of the latest tradeoff cuts. Thus, if a particular cut did eliminate the best-compromise solution, it would eventually be dropped, once again opening up the region containing the best-compromise solution. Still, this approach does not entirely alleviate the difficulties surrounding tradeoff approximations.

The problem of eliminating the best-compromise solution only manifests itself when a tradeoff cut is established for which this final solution is no longer feasible. In the TCP algorithm, decision maker supplied tradeoffs are used as a means of further restricting the decision space, not for approximating global preference functions

or establishing the best search direction. For this reason the importance of correctly assessing tradeoffs is somewhat reduced. As long as the tradeoffs are such that the corresponding tradeoff cut does not eliminate the best-compromise solution, the process is not in jeopardy. The difficulty is that one does not know a priori where the best-compromise solution is located. As a result, one is unable to ascertain whether or not a given tradeoff cut has eliminated this point.

The problem surrounding tradeoff sensitivity is compounded when one realizes that tradeoffs are only known for those points at which the algorithm has interacted with the decision maker. One is unable to artificially generate the decision maker's tradeoffs. This precludes the possibility of studying alternative paths leading to the final solution point.

In an effort to give the decision maker a better understanding of the latitude he is afforded with respect to tradeoffs, the Tradeoff Cutting Plane program (Appendix C) is designed to perform a tradeoff sensitivity analysis upon termination of the algorithm. For each tradeoff, upper and lower bounds are computed for which violation of these bounds would cause any succeeding point and/or the best-compromise solution to be eliminated or contradict an assumption of the algorithm.

The program assumes the last point generated by the algorithm is the best-compromise solution. It then calculates the amount one is able to perturb each tradeoff before forcing the best-compromise solution to become infeasible. Similarly, it also calculates what latitude is allowed before <u>any</u> succeeding point becomes infeasible.

Mathematically, the logic works as follows. Letting

$$w_i \equiv \frac{\partial U/\partial f_i}{\partial U/\partial f_1} ,$$

the tradeoff cut becomes

$$\sum_{i=1}^{r} w_i(f_i - f_i') \geq 0 ,$$

where f_i' is the value of the i^{th} objective at the current tradeoff point. Since the best-compromise must satisfy this cut,

$$\sum_{i=1}^{r} w_i(f_i^b - f_i') \geq 0 ,$$

where f_i^b is the value of the i^{th} objective at the best-compromise. For any given tradeoff w_j,

$$w_j(f_j^b - f_j') + \sum_{\substack{i=1 \\ i\neq j}}^{r} w_i(f_i^b - f_i') \geq 0 .$$

This establishes the following restrictions on w_j:

$$0 \leq w_j \begin{cases} \geq \dfrac{-\sum_{\substack{i=1 \\ i\neq j}}^{r} w_i(f_i^b - f_i')}{(f_j^b - f_j')} , & \text{if } (f_j^b - f_j') > 0 \\ \leq \dfrac{-\sum_{\substack{i=1 \\ i\neq j}}^{r} w_i(f_i^b - f_i')}{(f_j^b - f_j')} , & \text{if } (f_j^b - f_j') < 0 \\ \leq \infty , & \text{if } (f_j^b - f_j') = 0 . \end{cases}$$

Figure 3.5 illustrates what takes place when there are only two objective functions. Assuming $\underline{f}(\underline{x})$ to be the current tradeoff point, the objective space is partitioned into quadrants with $\underline{f}(\underline{x})$ being the point of reference. The computer routine bases all tradeoffs on a unit decrease in f_1 and assumes all other tradeoffs are fixed.

If the best-compromise is located in quadrant I, the upper and lower bounds are infinity and zero respectively. The reason for this stems from the fact that the decision maker's overall preference function is assumed to have a negative slope. Therefore, for a unit decrease in f_1, one is allowed to vary f_2 from zero to infinity without affecting the feasibility of the best-compromise. Equally straightforward is the case wherein the best-compromise is found in quadrant III. Any point located in this quadrant is necessarily infeasible. This being the case, it is impossible for quadrant III to contain the best-compromise.

If the best-compromise happens to land in quadrant IV, the results become more interesting. In decreasing f_1 by one unit, one determines the amount of increase allowed in f_2 before infeasibility occurs on the part of the best-compromise (see Figure 3.5). The lower bound on this perturbation is obviously zero; the upper bound is the maximum increase allowed before eliminating the best-compromise. An analogous result holds for the situation in which the best-compromise is located in quadrant II. Only this time one calculates a lower bound, and the upper bound is set at infinity.

An example of the program's tradeoff sensitivity printout is given in Figure 3.6. For each tradeoff point, the functional values

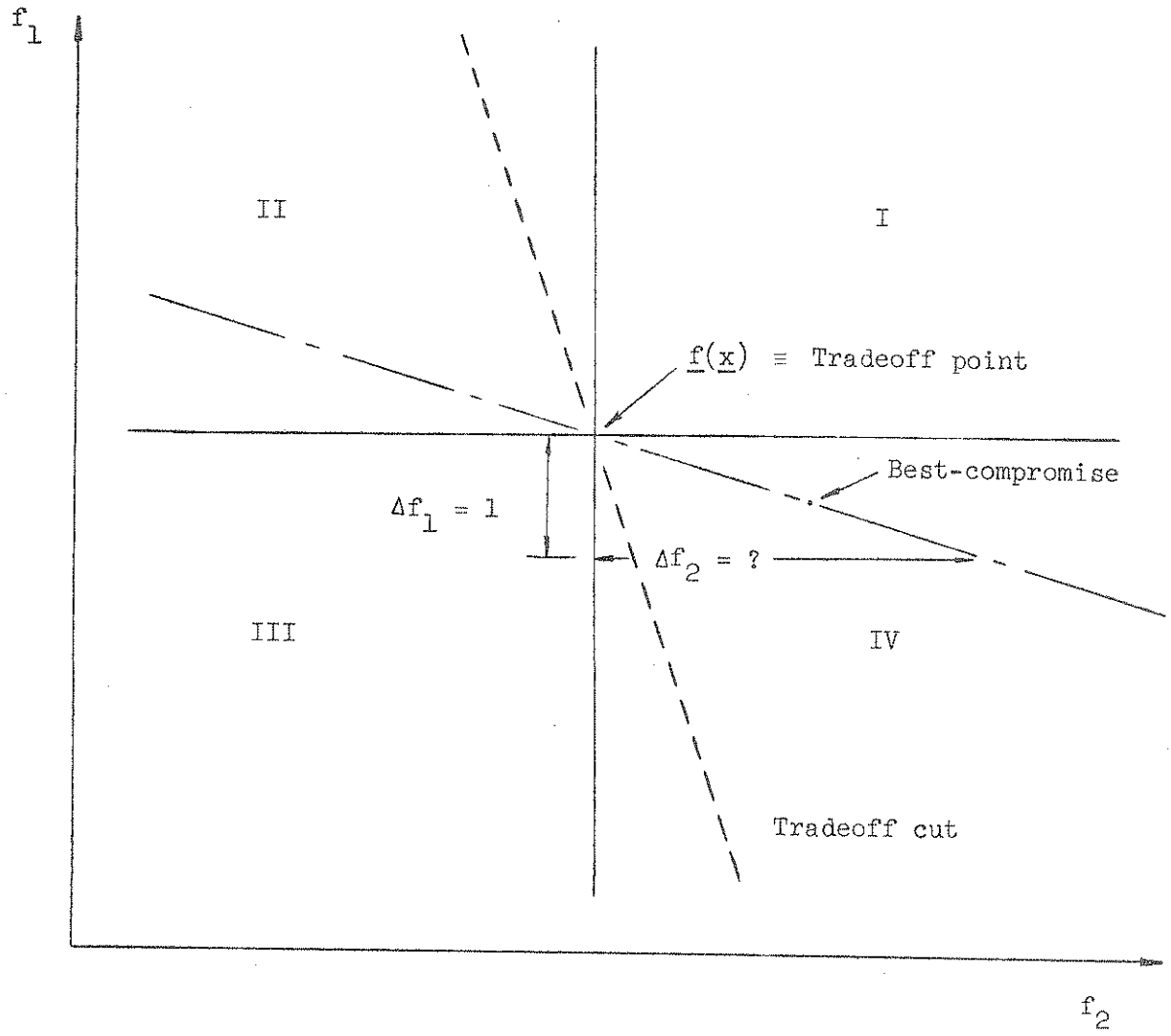


Figure 3.5. Tradeoff Sensitivity in the Case of Two Objectives

TRADEOFF SENSITIVITY BASED ON FINAL SOLUTION

TRADEOFF POINT 1 (F(1) = -6.598)

TRADEOFF	F	GIVEN	ALLOW (SUC PTS) MIN	ALLOW (SUC PTS) MAX	ALLOW (BC PT) MIN	ALLOW (BC PT) MAX
1 VS 2	-6.000	2.000E+01	3.396E-01	1.000E+30	3.396E-01	1.000E+30
1 VS 3	-5.707	5.000E-02	0	1.783E-01	0	2.998E-01
1 VS 4	-48.784	1.000E+00	2.829E-01	1.000E+30	1.758E-01	1.001E+30
1 VS 5	-9.197	1.000E+01	0	1.000E+30	0	1.001E+30

TRADEOFF POINT 2 (F(1) = -6.448)

TRADEOFF	F	GIVEN	ALLOW (SUC PTS) MIN	ALLOW (SUC PTS) MAX	ALLOW (BC PT) MIN	ALLOW (BC PT) MAX
1 VS 2	-12.633	5.000E+01	7.461E-03	1.000E+30	7.461E-03	1.000E+30
1 VS 3	-3.704	5.000E-02	1.721E-02	1.000E+30	1.107E-02	1.000E+30
1 VS 4	-60.156	1.000E-01	0	2.894E-01	0	4.544E-01
1 VS 5	-5.961	1.000E+01	3.054E-02	1.000E+30	0	1.000E+30

TRADEOFF POINT 3 (F(1) = -6.523)

TRADEOFF	F	GIVEN	ALLOW (SUC PTS) MIN	ALLOW (SUC PTS) MAX	ALLOW (BC PT) MIN	ALLOW (BC PT) MAX
1 VS 2	-9.316	1.000E+02	2.882E-01	1.000E+30	2.882E-01	1.000E+30
1 VS 3	-4.708	1.000E-01	1.218E-02	1.000E+30	0	1.000E+30
1 VS 4	-54.172	1.000E-01	0	5.402E-01	0	1.001E+30
1 VS 5	-6.822	1.000E+01	3.944E-02	1.000E+30	0	1.000E+30

TRADEOFF POINT 4 (F(1) = -6.560)

TRADEOFF	F	GIVEN	ALLOW (SUC PTS) MIN	ALLOW (SUC PTS) MAX	ALLOW (BC PT) MIN	ALLOW (BC PT) MAX
1 VS 2	-7.658	2.000E+01	4.091E-01	1.000E+30	4.091E-01	1.001E+30
1 VS 3	-5.206	5.000E-02	0	4.602E-01	0	4.602E-01
1 VS 4	-51.408	1.000E+00	1.180E-01	1.001E+30	1.180E-01	1.001E+30
1 VS 5	-7.714	1.000E+01	0	1.000E+30	0	1.001E+30

Figure 3.6. Tradeoff Sensitivity Printout

are printed together with the tradeoffs supplied by the decision maker. The values listed in the column entitled "given" are the tradeoffs supplied by the decision maker adjusted to correspond to a unit perturbation in f_1. The minimum and maximum perturbations allowed are given for both the best-compromise point and all points which follow the current tradeoff point (labeled "BC PT" and "SUC PTS" respectively). If the upper bound is infinity, the program is instructed to print 1.000E+30. In the case of an error (i.e., the best-compromise is located in quadrant III), the program prints -9.00E+00 for both the minimum and maximum values.

Under "TRADEOFF POINT 1" in Figure 3.6, f_1 and f_2 are shown to have values of -6.598 and -6.000 respectively. In addition, it shows the decision maker willing to trade 20 units of f_2 to gain 1 unit of f_1. Based on the final solution, the decision maker could have traded as little as 0.3396 of a unit of f_2 for the 1 unit gain in f_1, without eliminating any successive point (including the best-compromise point). As for f_3, it is shown to have a value of -5.707, for which the decision maker is willing to accept 1 unit of f_1 in trade for 0.05 of a unit of f_3. The decision maker could actually have traded as much as 0.1783 of a unit of f_3 without eliminating any of the successive points and as much as 0.2988 before eliminating the best-compromise point.

Practically speaking, care must be taken to understand what these limits mean. One should not assume that by using one of these extreme tradeoff values, the final solution will necessarily be the same. Indeed, even one change at a given tradeoff point would most likely

cause a different sequence of points to be followed. Yet, what these limits do represent is the degree of criticality associated with each tradeoff. If, for example, a given tradeoff is near the maximum allowable, one would realize what amount of tradeoff error would jeopardize the solution. So if a decision maker establishes a range within which a particular tradeoff is known to exist, this range can be judged against these limits to indicate to what extent his tradeoff indecision undermines the final result. An important fact to underscore is that in judging this acceptable margin of error, all other tradeoffs are assumed correct and fixed.

CHAPTER IV

STORM DRAINAGE - AN APPLICATION

In this chapter the Tradeoff Cutting Plane algorithm is used to analyze the expected storm drainage needs of an urban subbasin. Although a subbasin may be interesting from several water resource perspectives, this application focuses exclusively on the problem of storm drainage. Throughout the discussion, reference is made to the specific application of this analysis to the storm drainage problem associated with the city of West Lafayette, Indiana. First, a brief overview of this problem is given.

4.1. Overview

The medium size community of West Lafayette is able to be partitioned into a number of hydrologically independent watersheds. Each one may be viewed as having a separate drainage system composed of detention storage facilities, treatment plants, drainage networks, and a means of restitution into the receiving body of water. These watersheds can be further subdivided into subbasins. One such subbasin (number 13) is examined in terms of its storm drainage system. This subbasin is assumed, for the purposes of this analysis, to be hydrologically independent of the other subbasins, having its own drainage network, on-site detention storage facility, treatment plant, and tributary to a receiving water body.

The various factors involved in this problem as well as how they conceptually interlink are illustrated in Figure 4.1. Runoff due to rainfall and snowfall is channeled to a treatment facility before being discharged into the receiving body of water. That which is not able to be immediately treated is re-routed to a temporary storage facility to await treatment. Once the storage tank has filled, any further runoff either overflows into the receiving body of water, carrying with it pollutants, or backs-up, causing local flooding to occur.

Subbasin 13's storm drainage system is assumed to be characterized by three decision variables:

x_1 - local detention storage capacity (basin·inches),

x_2 - maximum treatment rate (basin·inches/hour), and

x_3 - maximum allowable overflow rate (basin·inches/hour).

Restricting the overflow rate makes it possible to incorporate into the analysis damages caused by local flooding. Once the detention storage facility is filled and the overflow rate is at its maximum, the drainage system's conveying capacity is significantly reduced. This results in local flooding, the consequence of which is structural damage and economic disruption to the area.

The hydrologic performance of subbasin 13 was examined by using an urban hydrologic simulation program termed LANDSTORM [14]. The program coupled a landuse forecasting model with an urban hydrologic simulation model called STORM [47, 56]. Based on such inputs as surface characteristics, growth requirements and zoning policies, the simulation program generated an urban growth scenerio for the

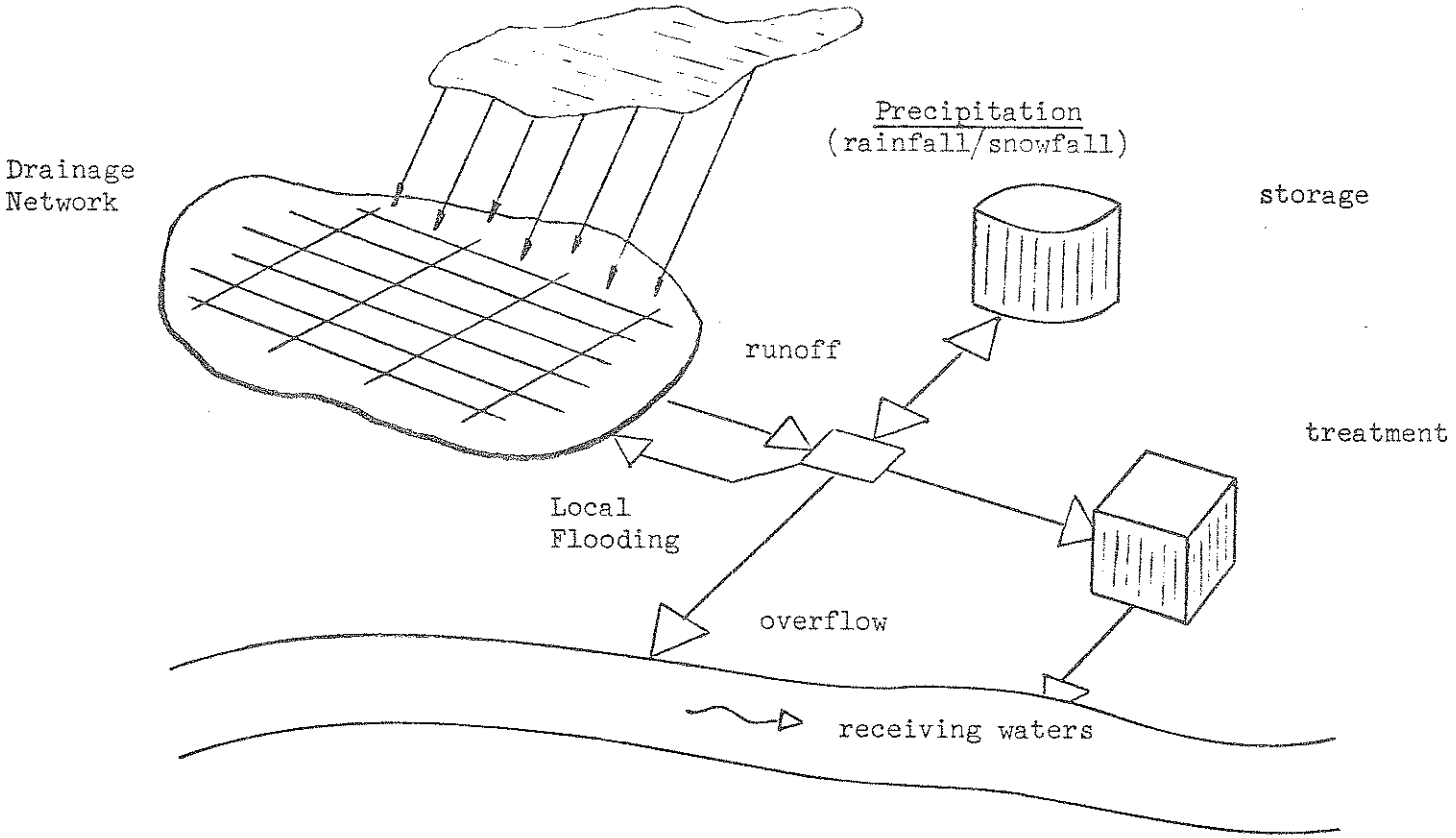


Figure 4.1. Conceptualization of Subbasin 13's Storm Drainage Problem

subbasin on a lot-by-lot basis. Following this, it simulated the performance of the given storm drainage system (i.e., (x_1,x_2,x_3)) over a specified period of time and under weather conditions representative of the area. Twenty-three years of recorded precipitation data from the West Lafayette area was used as input to the simulation. The model was then simulated for a 20 year period to obtain an estimate of the hydrologic performance of the subbasin's drainage system.

Output from the simulation consisted of the drainage system's yearly statistical performance. Relevant data included:

1. Average number of floods per year (y_1)
2. Average flood volumn per year (y_2)
3. Average number of pounds per year of suspended solids (y_{31})
4. Average number of pounds per year of settleable solids (y_{32})
5. Average number of pounds per year of biochemical oxygen demand - BOD (y_{33})
6. Average number of pounds per year of total nitrogen - N (y_{34})
7. Average number of pounds per year of ortho-phosphate - PO_4 (y_{35}).

4.2. Original Formulation

The original conceptualization, formulation, and analysis of this problem was done by Dr. Stergios Dendrou of the Civil Engineering Department of Purdue University. His comprehensive study [14] was of a much broader scope than the problem considered here. He investigated the larger scaled watershed drainage problem with its interdependent subbasins. The subbasins were not assumed to be necessarily hydrologically independent of each other. For example, situations were allowed for which the same treatment facility might service several subbasins. In order to coordinate the drainage of these subbasins, he used a multilevel optimization scheme. This scheme sought to minimize the cost of procuring, operating, and maintaining the watershed drainage system through the coordination of the individual subbasins. At the subbasin level, attention was paid to the detention storage capacities, treatment rates, and allowable overflow rates; while at the watershed level, interest turned to the allocation of these detention storage facilities and treatment plants among the various subbasins. By iterating between these two levels, the overall optimum was eventually identified.

4.3. Multiple Objective Formulation

In his study, Dendrou aggregated the various objectives into a single economic objective function. In doing so, he was implicitly assuming a constant rate of substitution among the objectives. That is, his local tradeoffs were invariable and global in that the same rate of substitution was applicable at any point in objective space. In the application described here, although limited to a single subbasin, the same "lower-level" storm drainage problem is considered,

but from a more representative, multiple objective standpoint. In fact, various sets of objectives are used as well as various trade-off policies between the objectives.

In Dendrou's model cost functions were representative of the storage and drainage network facilities, treatment plants, and damages (both structural and economic) caused by local flooding. Constraints on his system included hydraulic continuity equations, detention storage capacities, overflow threshold restrictions, and reliability conditions regarding pollutant concentrations during overflows, frequency of local floods, and street-flooding exceeding a threshold depth.

Only a summary of subbasin 13's analytical model is given here since details into the reasoning behind the model are well documented in Dendrou [14]. Table 4.1 lists each factor of interest together with its representative algebraic expression. Note that the objectives are now treated individually. All cost expressions take into account both the capital cost and the operating and maintenance cost appropriately discounted over the entire planning horizon.

Inasmuch as the basic submodel of this problem is a simulation program, it became necessary to approximate the behavior of the simulation output parameters by response surfaces. Over the region of interest,

$$0.01 \leq x_1 \leq 0.45$$

$$0.01 \leq x_2 \leq 0.10$$

$$0.01 \leq x_3 \leq 0.10,$$

Table 4.1. Analytical Model of Subbasin 13's Storm Drainage Problem

	FUNCTION	NAME	ALGEBRAIC EXPRESSION
Objectives	f_1	Drainage Network Cost	$106{,}780.37x_2 + 106{,}780.37x_3 + 61{,}704.67$
	f_2	Storage Facility Cost	$3{,}000x_1$
	f_3	Treatment Facility Cost	$(305{,}700/(.06 * 2289)^{.65})\ 2289x_2$
	f_4	Expected Flood Damage Cost	$250\ (2289)y_2$
	f_5	Expected Economic Loss Due to Flooding	$25{,}000y_1$
Constraints	g_1	Average Number of Floods per Year	y_1
	g_2	Probability of Exceeding a Flood Depth of 0.01 Basin-inches	$\frac{0.0000306}{x_1x_2} + 0.1082x_3 - 0.00986$
	g_{31}	Average Number of Pounds per Year of Suspended Solids	y_{31}
	g_{32}	Average Number of Pounds per Year of Settleable Solids	y_{32}
	g_{33}	Average Number of Pounds per Year of BOD	y_{33}
	g_{34}	Average Number of Pounds per Year of N	y_{34}
	g_{35}	Average Number of Pounds per Year of PO_4	y_{35}

a response surface was established for each of the seven output parameters listed in Section 4.1. Since the TCP algorithm assumes convexity, it was necessary to selectively choose regression models which not only satisfied this assumption but also allowed for a respectable fit. Numerous models were attempted for each of the seven parameters. In each case the model with the best "quality" fit (in terms of the coefficient of multiple determination R^2) was selected. The resulting multiple regression curves together with their corresponding R^2 values are given in Table 4.2.

To gain further insight into the quality of the fit associated with each expression found in Table 4.1, response surfaces were plotted over a two-dimensional plane (one variable being held fixed if necessary). Figures 4.2 and 4.3 show representative examples from a number of these plots. The top plot in each figure illustrates the results obtained from the storm drainage simulation model; the bottom plot shows the surface obtained by using the corresponding multiple regression curve.

4.4. Problem Statement

The multiple objective formulation of subbasin 13's storm drainage problem can now be explicitly stated as follows:

minimize

$$f_1 = 106{,}780.37x_2 + 106{,}780.37x_3 + 61{,}704.67$$

$$f_2 = 3{,}000x_1$$

$$f_3 = (305{,}700/(.06 \cdot 2289)^{.65})\ 2289x_2$$

$$f_4 = 250(2289)\ \exp(-39.75x_2 + 9.9x_3 + 2.74)$$

$$f_5 = 25\ \left(\frac{1.39}{x_1 x_2} + 4940x_3 - 80\right)$$

Table 4.2. Regression Models of the Simulation Output Parameters

PARAMETER	REGRESSION CURVE	R^2
y_1	$\frac{0.00139}{x_1 x_2} + 4.94x_3 - 0.08$	.921
y_2	$\exp(-39.75x_2 + 9.9x_3 + 2.74)$	.951
y_{31}	$\frac{12.307}{x_1 x_2} + 49{,}408.24x_3 - 4{,}051.02$	.906
y_{32}	$\frac{2.098}{x_1 x_2} + 8{,}046.33x_3 - 696.71$	.914
y_{33}	$\frac{2.138}{x_1 x_2} + 7{,}883.39x_3 - 705.04$	.910
y_{34}	$\frac{0.417}{x_1 x_2} + 1{,}721.26x_3 - 136.54$	.900
y_{35}	$\frac{0.164}{x_1 x_2} + 631.13x_3 - 54.48$	.912

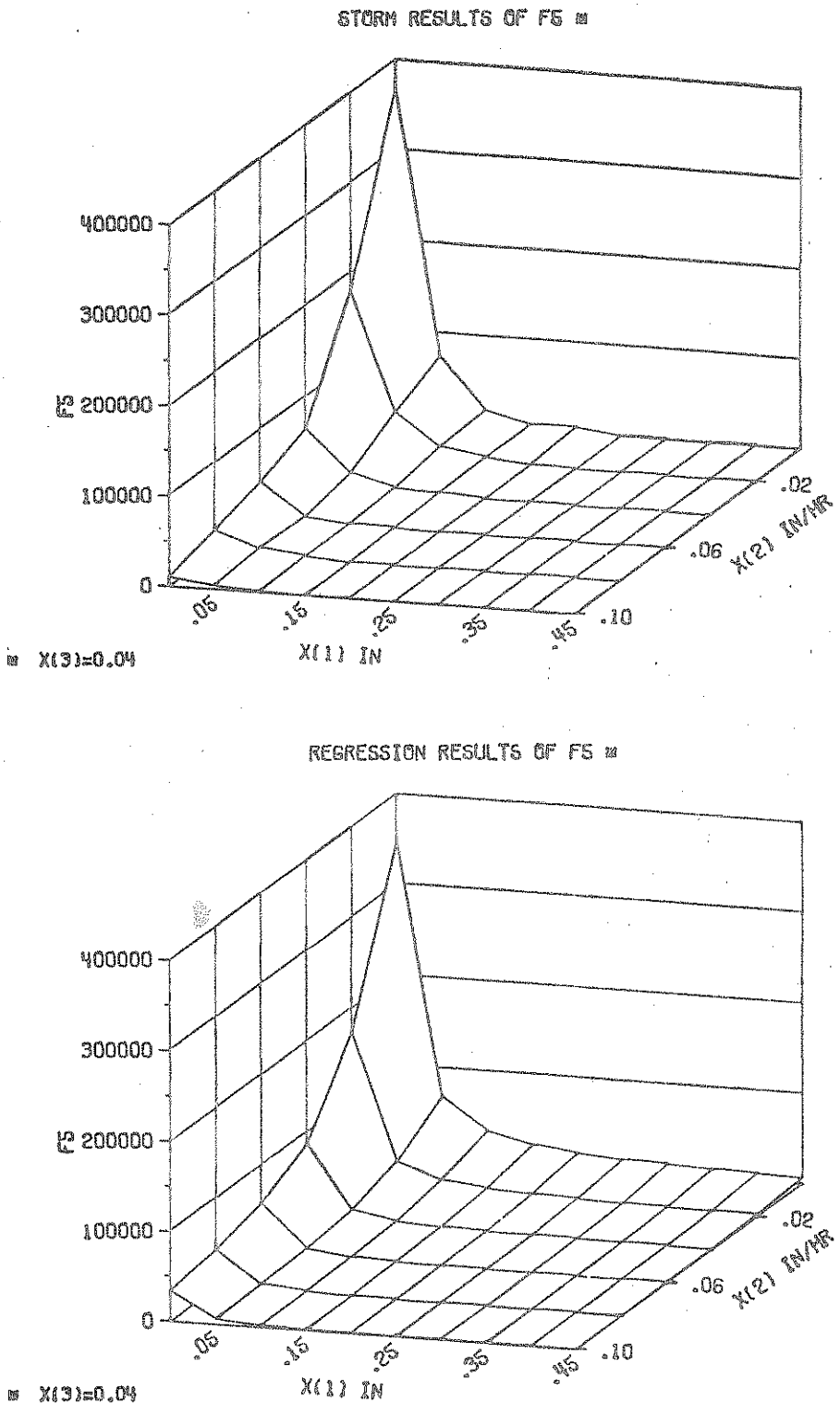


Figure 4.2. Response Surface and Regression Model of the Expected Economic Loss Due to Flooding

Table 4.2. Regression Models of the Simulation Output Parameters

PARAMETER	REGRESSION CURVE	R^2
y_1	$\frac{0.00139}{x_1 x_2} + 4.94x_3 - 0.08$	.921
y_2	$\exp(-39.75x_2 + 9.9x_3 + 2.74)$	.951
y_{31}	$\frac{12.307}{x_1 x_2} + 49{,}408.24x_3 - 4{,}051.02$	.906
y_{32}	$\frac{2.098}{x_1 x_2} + 8{,}046.33x_3 - 696.71$	.914
y_{33}	$\frac{2.138}{x_1 x_2} + 7{,}883.39x_3 - 705.04$	.910
y_{34}	$\frac{0.417}{x_1 x_2} + 1{,}721.26x_3 - 136.54$	.900
y_{35}	$\frac{0.164}{x_1 x_2} + 631.13x_3 - 54.48$	.912

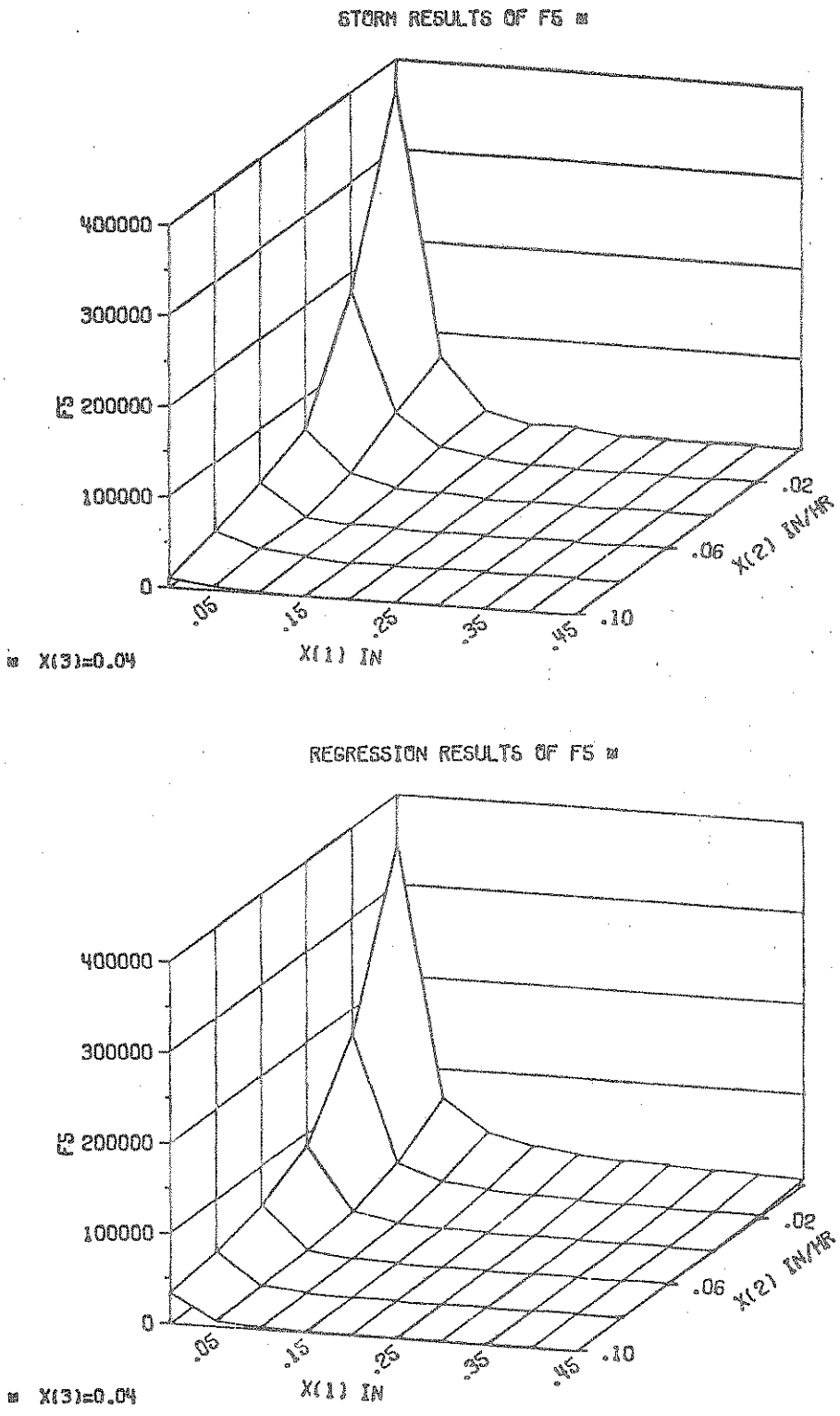


Figure 4.2. Response Surface and Regression Model of the Expected Economic Loss Due to Flooding

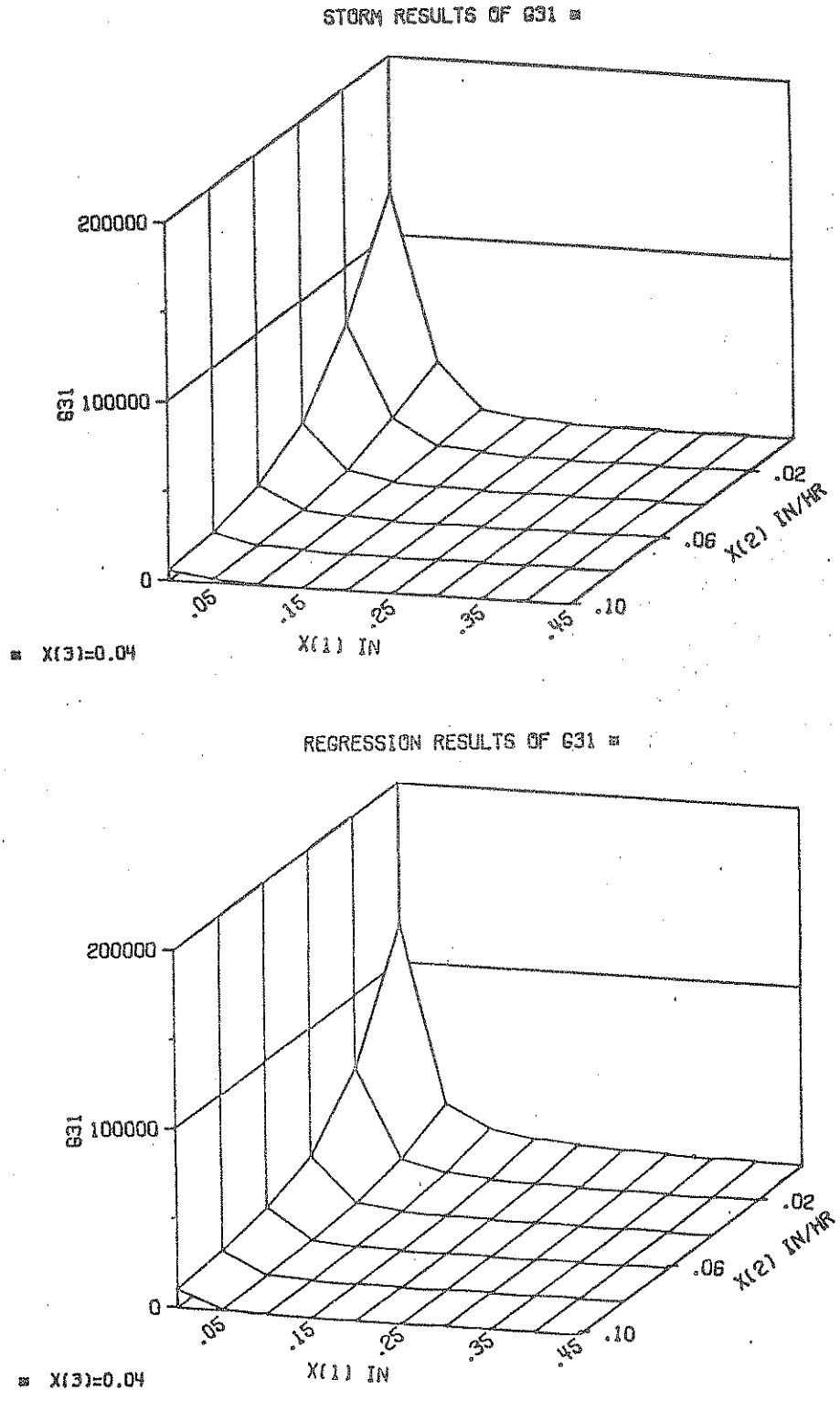


Figure 4.3. Response Surface and Regression Model of the Average Number of Pounds per Year of Suspended Solids

subject to

$$\frac{0.00139}{x_1 x_2} + 4.94x_3 - 0.08 \leq c_1$$

$$\frac{0.0000306}{x_1 x_2} + 0.1082x_3 - 0.00986 \leq c_2$$

$$\frac{12.307}{x_1 x_2} + 49408.24x_3 - 4051.02 \leq c_3$$

$$\frac{2.098}{x_1 x_2} + 8046.33x_3 - 696.71 \leq c_4$$

$$\frac{2.138}{x_1 x_2} + 7883.39x_3 - 705.04 \leq c_5$$

$$\frac{0.417}{x_1 x_2} + 1721.26x_3 - 136.54 \leq c_6$$

$$\frac{0.164}{x_1 x_2} + 631.13x_3 - 54.48 \leq c_7$$

$$0.01 \leq x_1 \leq 0.45$$

$$0.01 \leq x_2 \leq 0.10$$

$$0.01 \leq x_3 \leq 0.10 .$$

The values used for c_1 through c_7 are given in Table 4.3 along with the meaning of each constraint.

Table 4.3. Bounds Associated with Constraints g_1 through g_{35}

CONSTRAINT	MAXIMUM
Average number of floods per year	1
Probability of exceeding a flood depth of 0.01 basin·inches	0.10
Average number of pounds per year of suspended solids	50,000
Average number of pounds per year of settleable solids	16,000
Average number of pounds per year of BOD	10,000
Average number of pounds per year of N	2,000
Average number of pounds per year of PO_4	550

4.5. Tradeoff Policy A

The first tradeoff policy (A) used with the TCP algorithm in solving this multiple objective problem was a global policy. It reflected Dendrou's formulation of this problem for it aggregated the various objectives into a single function representative of the drainage system's total cost (Figure 4.4). This total sum objective was run primarily for comparison purposes. Since the single objective optimization problem can be solved using conventional methods, this policy allowed for a comparison of the solution with known results. Moreover, it gave an indication of the type of solution arrived at under Dendrou's formulation.

In order to study convergence characteristics and to allow for a check on the final solution, three separate starting points were used. These same three points were used with each policy so as to provide a means of studying the effect of policy changes. The three starting points were:

SP1 (.40,.01,.08),

SP2 (.20,.02,.02), and

SP3 (.30,.08,.01).

Each time, the algorithm was run for ten iterations, unless there was early termination due to the best-compromise solution being reached.

The optimal solution under policy A, found by applying the constrained minimization technique of Griffith and Stewart [23], is (0.4198, 0.0657, 0.0100) with a final objective function value of 2,594,869. Tables 4.4, 4.5, and 4.6 reveal the sequence of points

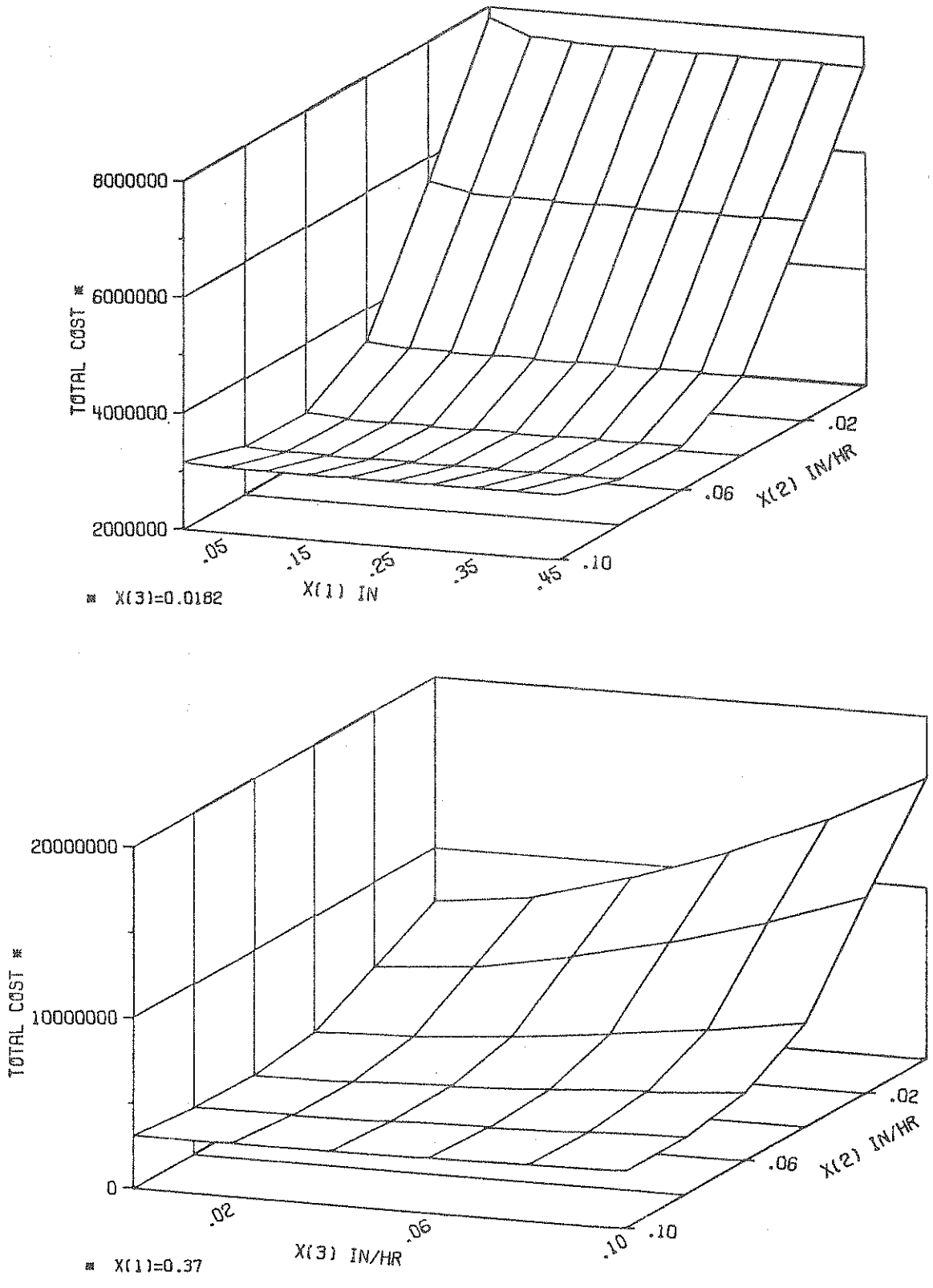


Figure 4.4. Total Cost of the Drainage System

Table 4.4. Sequence of Points Generated from SP1 under Policy A

Iteration	x_1	x_2	x_3	Total Cost	% Error
0	.4000	.0100	.0800	13,523,101	421.1
1	.3067	.0895	.0214	2,942,645	13.4
2	.3196	.0683	.0211	2,746,206	5.8
3	.3276	.0703	.0204	2,743,639	5.7
4	.3331	.0672	.0199	2,738,432	5.5
5	.3416	.0697	.0192	2,734,377	5.4
6	.3459	.0670	.0189	2,730,480	5.2
7	.3526	.0693	.0183	3,727,256	5.1
8	.3561	.0669	.0180	2,724,145	5.0
9	.3616	.0689	.0176	2,721,493	4.9
10	.3644	.0668	.0173	2,718,939	4.8

Table 4.5. Sequence of Points Generated from SP2 under Policy A

Iteration	x_1	x_2	x_3	Total Cost	% Error
0	.2000	.0200	.0200	5,524,845	112.9
1	.3316	.0895	.0222	2,945,373	13.5
2	.3363	.0684	.0216	2,750,223	6.0
3	.3437	.0705	.0209	2,747,224	5.9
4	.3485	.0672	.0204	2,741,514	5.7
5	.3561	.0699	.0196	2,737,138	5.5
6	.3598	.0670	.0192	2,732,921	5.3
7	.3657	.0694	.0186	2,729,444	5.2
8	.3687	.0669	.0183	2,726,118	5.1
9	.3735	.0690	.0178	2,723,316	5.0
10	.3759	.0668	.0176	2,720,586	4.8

Table 4.6. Sequence of Points Generated from SP3 under Policy A

Iteration	x_1	x_2	x_3	Total Cost	% Error
0	.3000	.0800	.0100	2,762,576	6.5
1	.3159	.0668	.0116	2,676,679	3.2
2	.3179	.0659	.0116	2,676,250	3.1
3	.3194	.0663	.0115	2,676,163	3.1
4	.3204	.0659	.0115	2,676,058	3.1
5	.3220	.0663	.0115	2,675,957	3.1
6	.3230	.0659	.0115	2,675,857	3.1
7	.3245	.0663	.0115	2,675,755	3.1
8	.3254	.0659	.0115	2,675,661	3.1
9	.3268	.0663	.0115	2,675,570	3.1
10	.3277	.0659	.0115	2,675,480	3.1

generated by the TCP algorithm from each starting point. Also listed is, for each point in the sequence, the total cost and the amount of error from the optimal solution (i.e.,

$$\% \text{ Error} = \frac{\text{Total Cost} - 2,594,869}{2,594,869} \cdot 100\%).$$

For each starting condition, Figure 4.5 shows the total cost as a function of the iteration number. Notice from this figure the relatively large initial improvement in the total cost function. One quickly moves into the neighborhood of the true solution after only three iterations. From a practical standpoint this is significant for decision makers are typically unwilling, under a man-machine interactive environment, to spend much time searching for the best solution.

4.6. Tradeoff Policy B

Turning to the second tradeoff policy (B), an effort was made to once again consider a global policy, but this time make it dependent upon the values of the objectives at the various local tradeoff points. The policy stated that the amount one was willing to trade of a given objective for a unit decrease in f_1 was considered inversely proportional to a percentage value of the objective. In other words, if a given objective had a high cost in comparison to f_1, the decision maker favored (in reverse proportion to their values) increasing the cost associated with f_1 over that of the given objective.

Under this policy, one finds again this rather rapid stabilization in the values of the objectives. As seen in Figures 4.6 through 4.10, all five objectives have begun to dampen their oscillation pattern

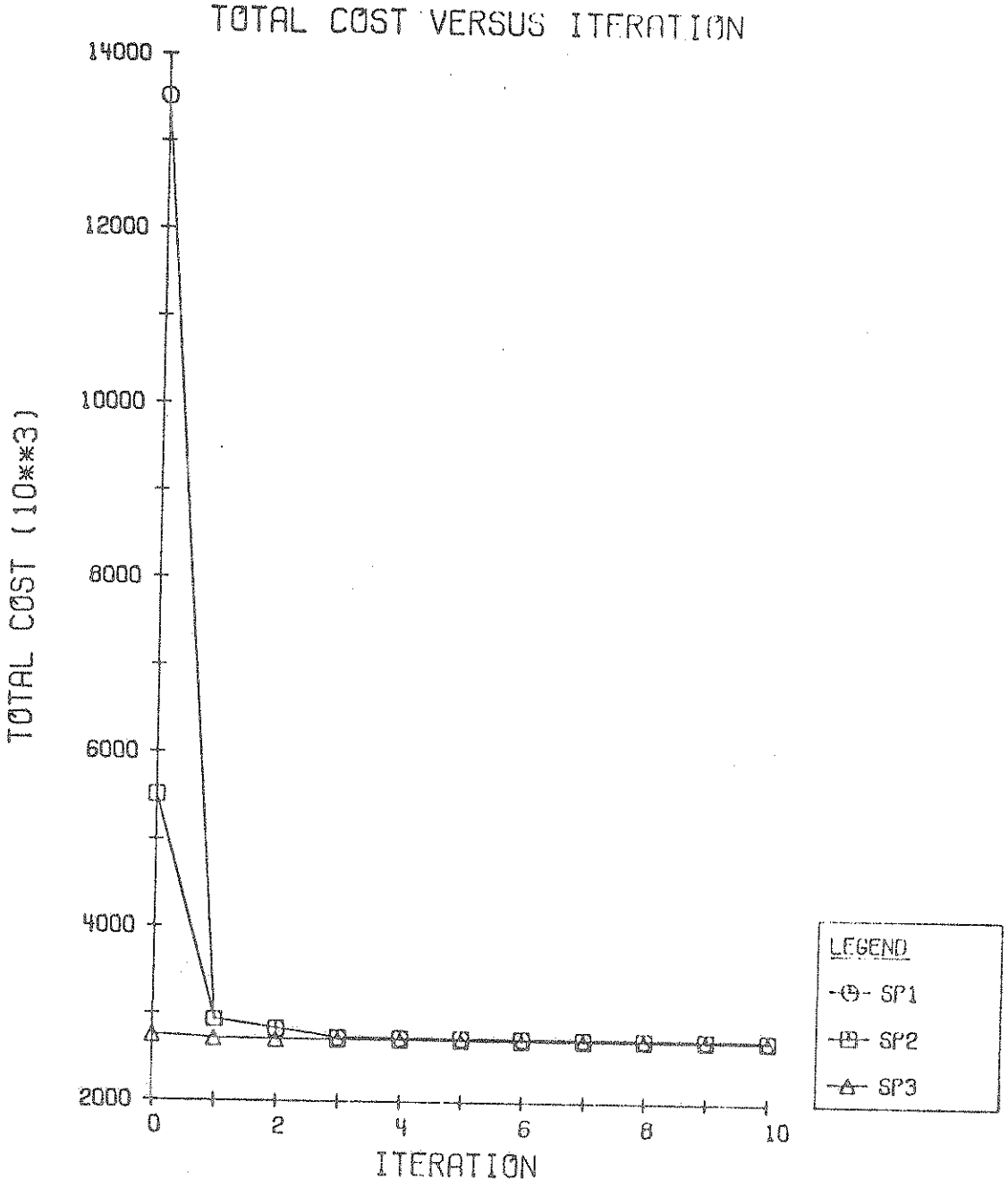


Figure 4.5. Drainage System Cost as a Function of the Iteration Number

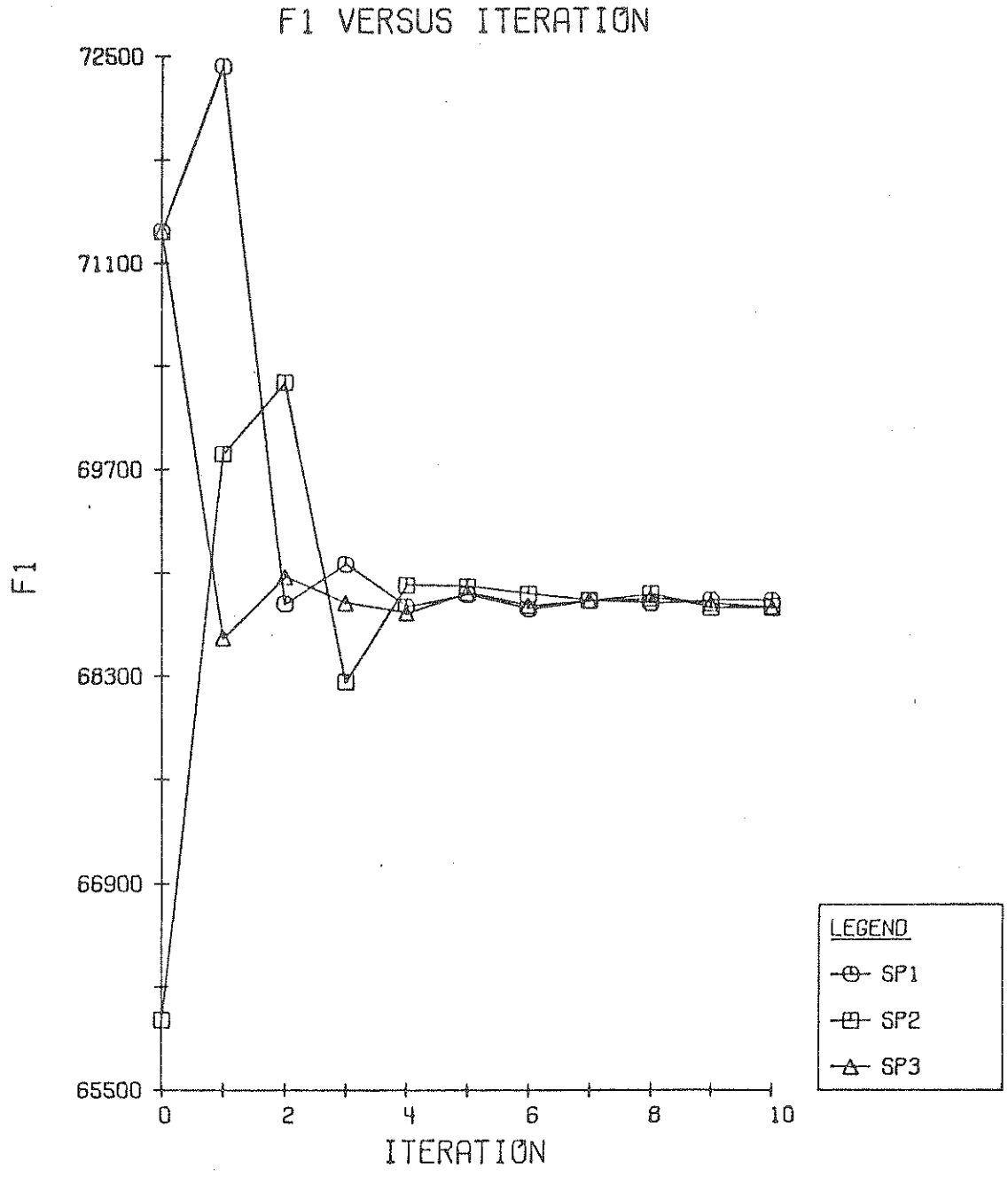


Figure 4.6. Drainage Network Cost Under Policy B

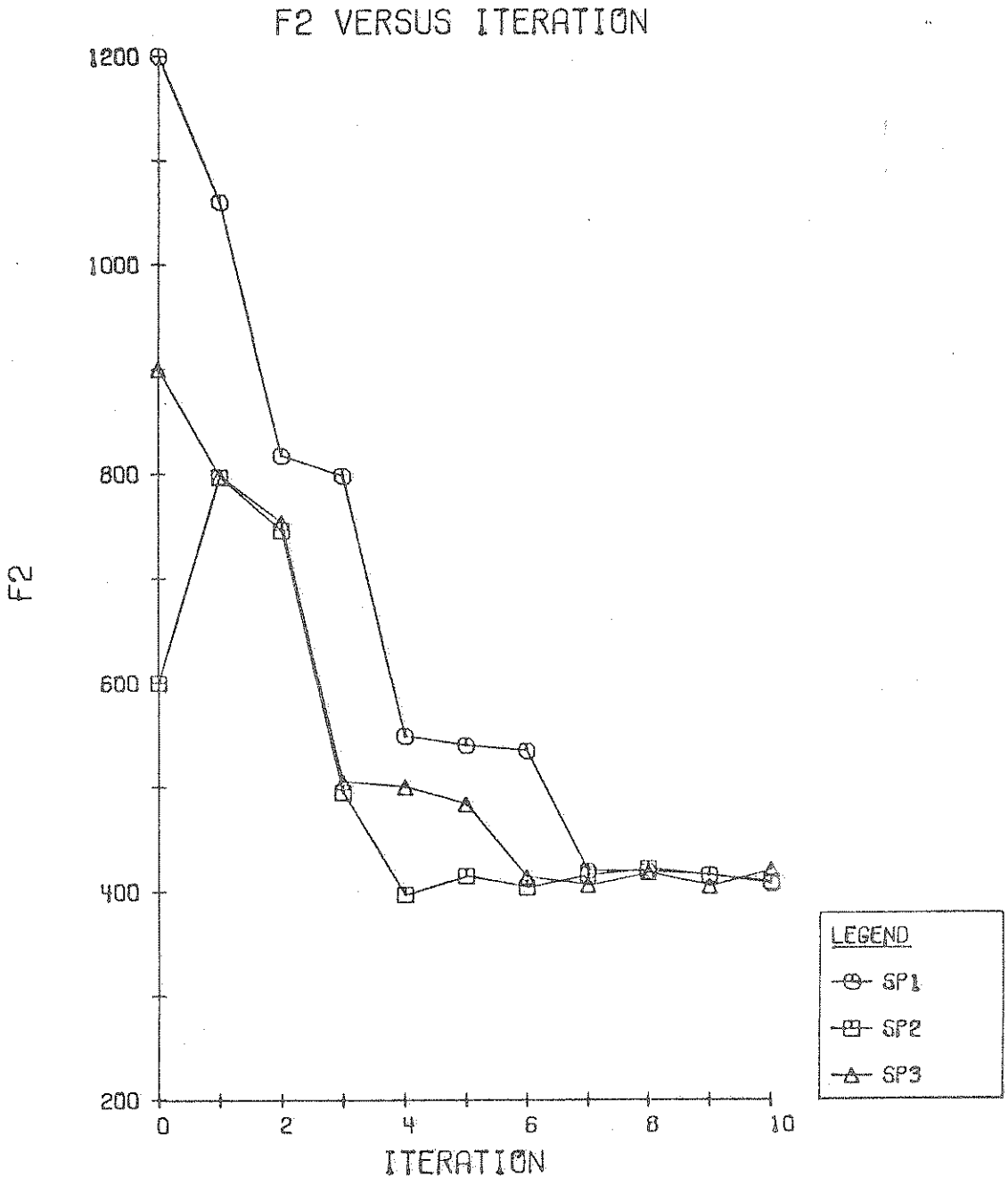


Figure 4.7. Storage Facility Cost Under Policy B

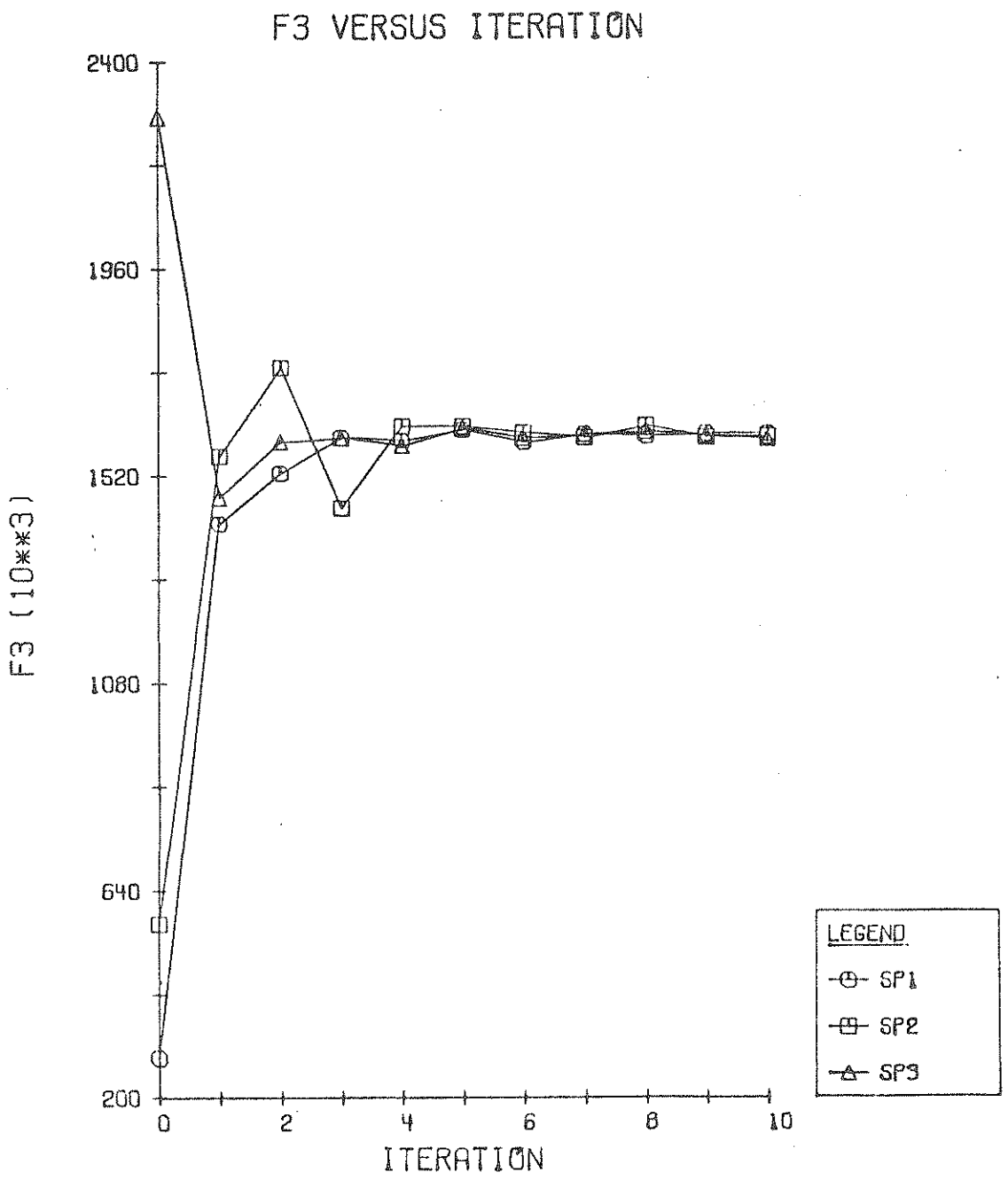


Figure 4.8. Treatment Facility Cost Under Policy B

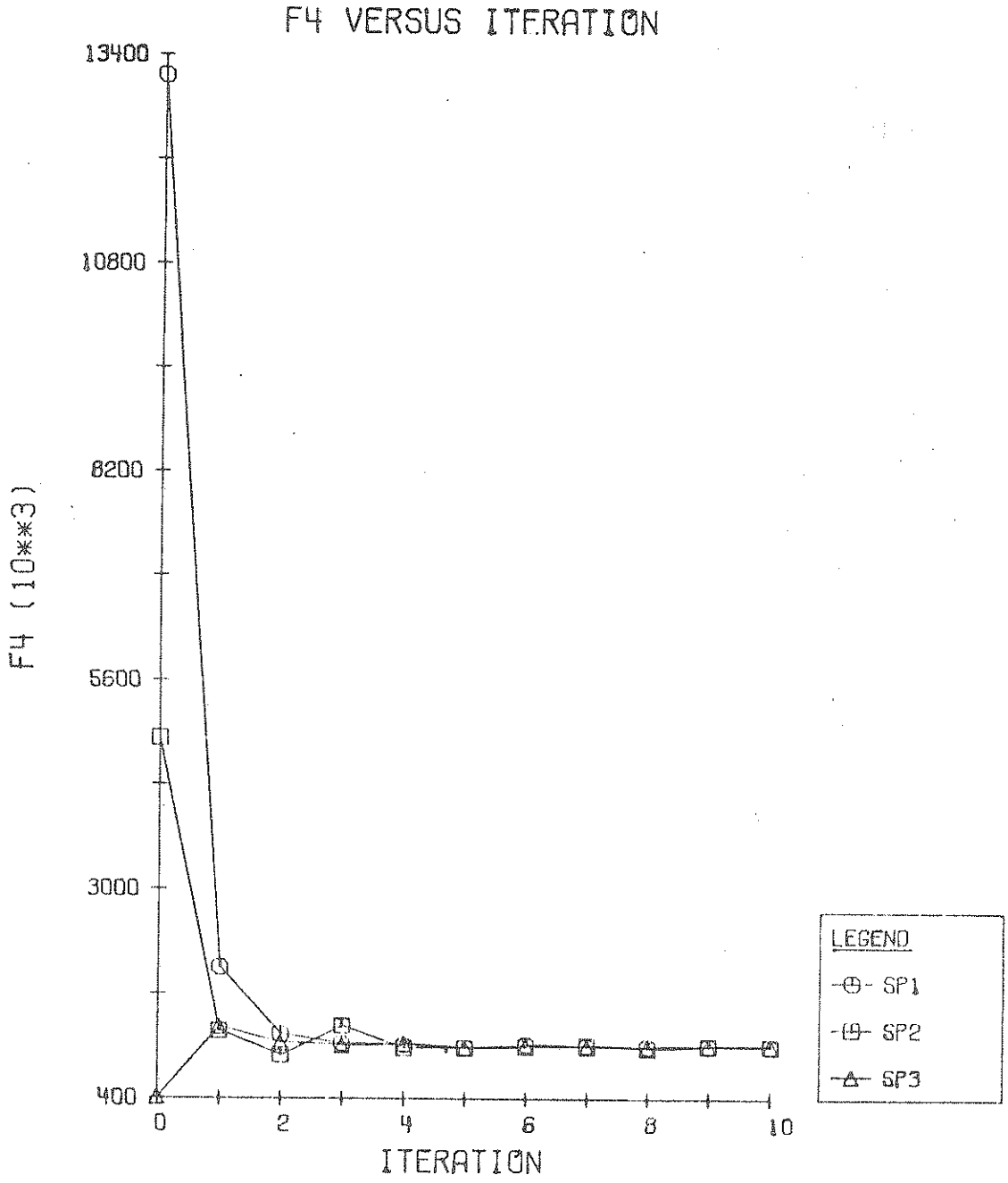


Figure 4.9. Expected Flood Damage Cost Under Policy B

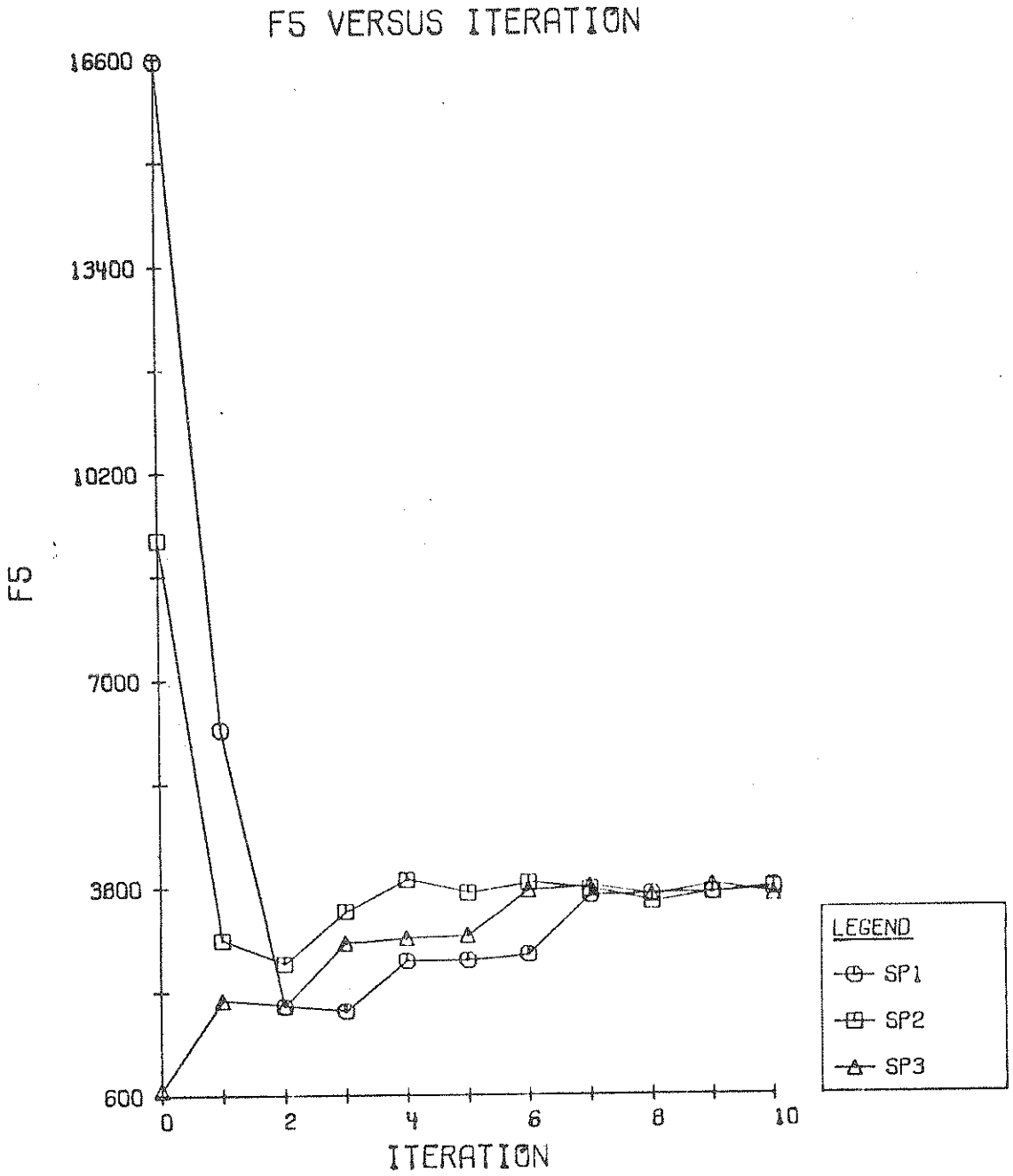


Figure 4.10. Expected Economic Loss Due to Flooding Under Policy B

after only seven iterations. The best-compromise level of each objective function is quickly sought irrespective of the starting condition. Tables 4.7 through 4.9 list the sequence taken from each starting point.

4.7. Tradeoff Policy C

The third tradeoff policy (C) retained the intrinsic value associated with each of the five objectives and used a decision maker to assess the local tradeoffs. Dr. Dendrou, the original modeler of the system, supplied the necessary information. He served as a well-informed decision maker, fully understanding the intricate characteristics of the drainage system as well as the type of information required on each iteration of the algorithm.

Policy C's sequence of points for each starting condition is given in Table 4.10. Notice that the local tradeoffs supplied by the decision maker resulted in inconsistent solutions. x_1 and x_3 tended to converge to the same level, whereas x_2 ranged from .0146 to .0829. Comments concerning this type of discrepancy will be made shortly. Observe also that the results here do not coincide with those under policy A. Apparently, the total cost objective under policy A does not resemble the decision maker's preference function represented under policy C. The decision maker failed to follow (as is his preference) the same pattern of behavior expected of a predominately cost conscious person. Influences beyond the obvious monetary ones swayed his tradeoff assessments.

4.8. Tradeoff Policy D

A reformulation of the multiple objective problem accompanied this final tradeoff policy (D). The cost functions f_1, f_2, and f_3

Table 4.7. Sequence of Points Generated from SP1 under Policy B

Iteration	x_1	x_2	x_3	f_1	f_2	f_3	f_4	f_5
0	.4000	.0100	.0800	71,315	1,200	285,347	13,148,671	16,568
1	.3533	.0498	.0507	72,431	1,060	1,419.765	2,025,718	6,238
2	.2724	.0535	.0129	68,788	817	1,525,639	1,201,929	1,975
3	.2659	.0561	.0127	69,057	798	1,601,739	1,079,472	1,899
4	.1829	.0559	.0103	68,773	549	1,594,551	1,064,614	2,673
5	.1800	.0566	.0103	68,849	540	1,615,202	1,034,369	2,683
6	.1785	.0557	.0103	68,754	535	1,589,800	1,071,605	2,767
7	.1400	.0563	.0102	68,809	420	1,606,687	1,045,876	3,672
8	.1395	.0562	.0102	68,792	419	1,602,329	1,052,230	3,698
9	.1387	.0563	.0102	68,812	416	1,607,697	1,044,359	3,710
10	.1356	.0563	.0102	68,807	407	1,606 631	1,045,804	3,812

Table 4.8. Sequence of Points Generated from SP2 under Policy B

Iteration	x_1	x_2	x_3	f_1	f_2	f_3	f_4	f_5
0	.2000	.0200	.0200	65,976	600	570,694	4,878,417	9,158
1	.2658	.0548	.0211	69,805	797	1,562,474	1,238,804	2,994
2	.2486	.0613	.0190	70,287	746	1,750,064	934,682	2,631
3	.1651	.0509	.0105	68,258	495	1,452,116	1,300,488	3,431
4	.1323	.0569	.0106	68,912	397	1,624,468	1,023,729	3,919
5	.1384	.0569	.0106	68,907	415	1,623,593	1,024,861	3,717
6	.1348	.0564	.0105	68,854	404	1,609,902	1,044,363	3,871
7	.1385	.0560	.0105	68,810	416	1,599,191	1,059,776	3,773
8	.1406	.0569	.0100	68,853	422	1,624,527	1,018,067	3,577
9	.1385	.0560	.0100	68,758	415	1,599,151	1,054,703	3,714
10	.1364	.0561	.0100	68,768	409	1,601,883	1,050,694	3,776

Table 4.9. Sequence of Points Generated from SP3 under Policy B

Iteration	x_1	x_2	x_3	f_1	f_2	f_3	f_4	f_5
0	.3000	.0800	.0100	71,315	900	2,282,775	406,903	683
1	.2661	.0517	.0125	68,557	798	1,475,441	1,283,850	2,065
2	.2514	.0558	.0122	68,971	754	1,592,982	1,087,307	1,985
3	.1682	.0561	.0103	68,793	505	1,600,790	1,055,084	2,952
4	.1667	.0555	.0103	68,726	500	1,583,111	1,081,358	3,027
5	.1613	.0567	.0103	68,860	484	1,619,196	1,028,250	3,065
6	.1380	.0560	.0102	68,777	414	1,598,297	1,058,135	3,758
7	.1355	.0563	.0102	68,807	406	1,606,550	1,045,978	3,817
8	.1393	.0565	.0102	68,822	418	1,611,270	1,038,851	3,675
9	.1352	.0562	.0102	68,787	405	1,602,494	1,051,399	3,834
10	.1402	.0559	.0101	68,758	421	1,595,896	1,060,670	3,683

Table 4.10. Sequence of Points Generated under Policy C

Iteration	SP1			SP2			SP3		
	x_1	x_2	x_3	x_1	x_2	x_3	x_1	x_2	x_3
0	.4000	.0100	.0800	.2000	.0200	.0200	.3000	.0800	.0100
1	.3805	.0266	.0678	.4211	.0130	.0130	.3215	.0829	.0101
2	.3887	.0161	.0161	.4347	.0157	.0114	.4496	.0829	.0100
3	.3911	.0195	.0159	.4464	.0149	.0104	.4500	.0829	.0100
4	.3931	.0212	.0157	.4489	.0147	.0101	.4500	.0829	.0100
5	.3948	.0227	.0155	.4497	.0146	.0100		*	
6	.3969	.0245	.0153	.4499	.0146	.0100			
7	.4460	.0217	.0104	.4499	.0146	.0100			
8	.4496	.0213	.0100	.4500	.0146	.0100			
9	.4496	.0213	.0100	.4500	.0146	.0100			
10	.4496	.0213	.0100		*				

* Termination due to best-compromise solution being reached

were added to form a single objective, as were the local flooding costs f_4 and f_5. Furthermore, the expressions reflecting pollutant loads (i.e., constraints g_{31}, g_{32}, g_{33}, g_{34}, g_{35}) were dropped as constraints and reformulated to function as objectives. Each pollutant type was represented by a separate objective function. Once again, the aim was to minimize the set of objectives under the local trade-off preferences of Dr. Dendrou.

Policy D's performance is given in Table 4.11. It shows, for each starting point, the path taken to the best-compromise solution.

4.9. Discussion

For comparison purposes, let the solution to each policy be represented by the following derived answers:

Policy A - (.3277, .0659, .0115),

Policy B - (.1364, .0561, .0100),

Policy C - (.4500, .0829, .0100),

Policy D - (.3508, .0666, .0111).

Looking at the differences in these solutions, one quickly recognizes that the maximum allowable overflow rate (x_3) is the least significant variable across policies. In each case the overflow rate was forced to its minimum or near-minimum level. Note also the similar results obtained under policies A and D. Evidently, the inclusion of pollutant loads as objectives (as well as using a decision maker) failed to cause any notable differences. This result is contrary to the one observed earlier between policies A and C wherein the inclusion of the decision maker did cause discrepancies between the two policies.

Table 4.11. Sequence of Points Generated under Policy D

Iteration	SP1			SP2			SP3		
	x_1	x_2	x_3	x_1	x_2	x_3	x_1	x_2	x_3
0	.4000	.0100	.0800	.2000	.0200	.0200	.3000	.0800	.0100
1	.2300	.0320	.0780	.2299	.0320	.0780	.3155	.0682	.0112
2	.2300	.0484	.0780	.2300	.0484	.0780	.3191	.0666	.0112
3	.2300	.0837	.0780	.2300	.0837	.0780	.3257	.0683	.0112
4	.2624	.0673	.0631	.2624	.0673	.0631	.3291	.0666	.0112
5	.3401	.0910	.0160	.3401	.0910	.0160	.3346	.0682	.0112
6	.3394	.0696	.0176	.3394	.0696	.0176	.3374	.0666	.0112
7	.3439	.0669	.0173	.3438	.0669	.0173	.3421	.0681	.0111
8	.3563	.0713	.0165	.3563	.0713	.0165	.3446	.0666	.0111
9	.3611	.0672	.0163	.3611	.0672	.0163	.3486	.0680	.0111
10	.3591	.0656	.0157	.3593	.0659	.0157	.3508	.0666	.0111

It should be emphasized that the TCP algorithm is not confined to just local tradeoff policies. Although the algorithm was developed on the premise that local tradeoffs would be given by the decision maker, this does not preclude the use of this algorithm in situations involving global tradeoff policies, for knowledge of one's global preference function implies an understanding of local behavior. Moreover, in requiring these local tradeoff assessments, considerably less is demanded of the decision maker than if he were asked to be definitive about his global preference function.

Finally, a comment about including the human element in the solution process. As was illustrated under policy C, inconsistencies can arise when a decision maker is asked to supply the necessary information to effect a solution. The inconsistency shown by the decision maker in starting from different locations in decision space exposes the fact that he does not always behave according to a unique overall preference function. Reduction of some of this erratic behavior is usually accomplished by performing tradeoff consistency checks at the various iteration points. Testing consistency at a single point is a standard procedure with tradeoff assessment schemes [15, 22, 34, 43]. It involves using a different reference criterion and the chain rule to reassess the tradeoffs. Because there can be only $r - 1$ unique tradeoffs among the objectives, one is able to measure the disagreement connected with each tradeoff. A predetermined tolerance level is then used to ascertain whether or not it is necessary to interact any further with the decision maker.

Experience has shown that decision makers tend to be cognizant of only a few objectives at any one time. Those objectives which

exhibit unusual or critical values are inclined to receive the most attention. Consequently, the decision maker might be more conscious of certain objectives at one location than at another. This phenomenon is reflected in the tradeoffs the decision maker assesses between these objectives and, thus, causes his overall preference function to deviate according to the starting point.

Often, by starting the interactive solution process from different starting points, certain decision variables repeatedly seek the same level while others seem to disguise their true intentions. Under tradeoff policy C, both x_1 and x_3 sought the same levels each time, while x_2 never did. Although the human element may be the cause of such behavior, it should not be viewed as a destructive influence. By incorporating these human inconsistencies into the decision process, one is able to recognize those decision variables having rather robust solution levels. Further analysis of the problem can then be carried out with these variables fixed at their appropriate levels. In fact, having a condensed version of the original problem might even enhance the chances of better judgements on behalf of the decision maker.

This last variable reduction technique reflects the type of thinking sometimes found in the statistical analyses of a process. Being able to identify those variables which do not significantly affect the process is quite often helpful in resolving the problem. In fact, the TCP algorithm, when used in this way, might prove to be a very useful means of further refining the original problem.

CHAPTER V

DISCRETE OPTIMIZATION VIA THE TRADEOFF CUTTING PLANE ALGORITHM

In this chapter the TCP algorithm is modified in order to solve discrete multiple objective mathematical programming problems. The direct use of the former algorithm is prohibited by the inherent nature of this type of problem. The chapter begins with a general interpretation of the discrete version of the TCP algorithm.

5.1. General Approach

The optimization method developed in this chapter, being gradient-based, works with the continuous feasible space X defined in Section 1.5. The solution, however, must be found in the set X_I which consists of those points in X which have integral valued components. The set X_I is assumed to contain at least one lattice point. U and f_i are still assumed to satisfy (with respect to the feasible set X) all the properties previously specified in Section 1.5. To be practical, interaction with the decision maker is only allowed at points in X_I. Also, in an effort to simplify the presentation, the expression best-compromise set is introduced.

Definition. (Best-Compromise Set)

The best-compromise set is the collection of all best-compromise solutions. In set-builder form, it is

$$\{\underline{z} \mid U(\underline{z}) \geq U(\underline{x}), \quad \forall\ \underline{x} \in X_I ; \quad \text{and } \underline{z} \in X_I \}.$$

In addressing the problem of discrete multiple objective optimization, the basic approach is slightly modified from what has previously been explained with regards to continuous solutions, but the basic concept of shrinking the original decision space by means of tradeoff cuts remains germane to the process. Instead of locating a continuous solution on each iteration, an integer solution is selected to act as the next iteration point. By establishing the tradeoff cut, the decision space is further restricted. Continuing in this manner reduces the original decision space to a set of integer points (or a single integer point) which contains the best-compromise solution.

Although the best-compromise solution is contained in this final integer point set, it is possible for this final set to contain integer points other than the best-compromise solution. At first this may appear discouraging. Upon closer examination, however, one learns that this final set is "nearly" identical to the best-compromise set.

Exactly how "close" this set is to being the best-compromise set is the subject of some later propositions. Before investigating this relationship, the TCP algorithm for discrete optimization will be discussed in more detail.

5.2. Development of the Algorithm

The TCP algorithm starts with a feasible integer solution. This point is termed the initial iteration point and is designated $\underline{x}^{\circ}$. Upon interacting with the decision maker to determine his local functional tradeoffs, a tradeoff cut is established at $\underline{x}^{\circ}$. In contrast to what

is done in the continuous case, these local tradeoffs play a direct role in determining the direction to improve the overall preference function U. Similar to the Geoffrion-Dyer-Feinberg approach, the gradient direction of U is used. This means that local tradeoffs $\underline{t}^{\circ}$ combine with the gradients of each objective function f_i to form a linear approximation of the overall preference function U. This approximation of U (viz., $\alpha \nabla_{\underline{x}} U(\underline{x}^{\circ})^{t} \cdot \underline{x}$) is used as the objective function of an integer programming (IP) problem. Since the solution of an IP problem is invariant to positive multiplications of the linear objective function, using this positive multiple of U's linear approximation does not affect the outcome.

The purpose of the integer programming problem is to locate the next iteration point. The linear approximation of U is maximized over the integer points contained in the original feasible set intersected with the convex set defined by the translated tradeoff cut. As a result, the next iteration point is discrete and feasible. The TCP algorithm automatically designates each iteration point as a "potential point". Its classification as a potential point continues until it fails to satisfy a tradeoff cut.

Definition. (Potential Point)

A potential point is any integer point $\underline{x} \in X_I$ which has been generated by the Tradeoff Cutting Plane algorithm and which currently satisfies every tradeoff cut.

Definition. (Potential Set)

The collection of all potential points forms the potential set.

At each new iteration point, the IP problem is again formed. Upon solving it, local tradeoffs are again asked of the decision maker. A check is made to see which potential points (i.e., past iteration points which have remained feasible) violate the present tradeoff cut. Observe that the current iteration point will never violate its own cut, so attention is only paid to potential points other than the current one. For this example the initial iteration point, considered a potential point by definition, is now tested to see if it violates the current tradeoff cut. If it violates the cut it is no longer considered a member of the potential set and, consequently, no longer a candidate to become a best-compromise solution.

Figure 5.1 shows how iteration points are eliminated and the way in which the decision space is continually reduced in size. After receiving the decision maker's tradeoffs at point $\underline{f}(A)$, the translated tradeoff cut is superimposed on the original convex decision space to form a new, more stringent, feasible region. The linear approximation of U at point A is then used to find the new (integer) iteration point. This iteration point (point B) is labeled a potential point, meaning that it is now a candidate for the best-compromise solution. Notice that the decision maker's tradeoffs at $\underline{f}(B)$ result in a translated tradeoff cut which eliminates point A, a former potential point. Taking the linear approximation of U at B and solving the associated IP problem produces the next potential point (point C).

Ideally, this process will continue to locate new improved potential points and eliminate past poor selections until the

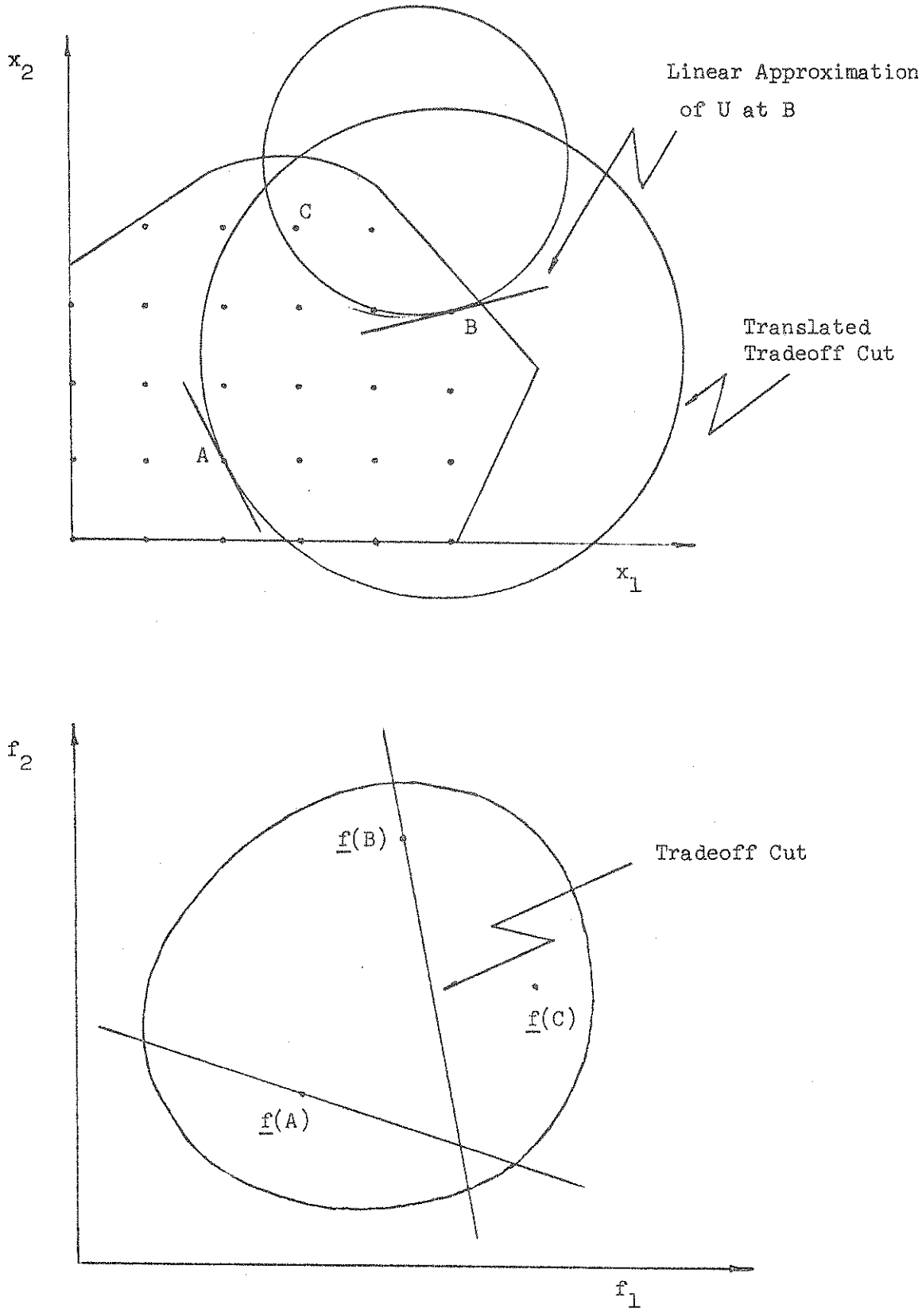


Figure 5.1. Tradeoff Cut Elimination Scheme

updated feasible region contains only one integer point, that being the best-compromise solution. Unfortunately, this cannot be guaranteed.

Taking the example one step further, Figure 5.2 illustrates the type of difficulty that is encountered. The linear approximation of U at B has generated point C. The tradeoff cut at $\underline{f}(C)$ has not eliminated $\underline{f}(B)$. Consequently, the translated tradeoff cut at point C does not eliminate point B. Using a linear approximation at C causes the IP problem to return point B as a solution without any consideration being given to other, possibly superior, interior points. As a result, cycling occurs between these two potential points, preventing the best-compromise solution from ever being realized.

To counteract this behavior, a strategy must be devised which will exclude a point from further consideration if it has previously been designated as a potential point. Fortunately, a technique [33] exists. Its approach is to literally circumvent the potential point by adding constraints. The method is particularly well suited to this problem of cycling for it allows elimination of the potential points without affecting the eligibility of the other feasible solutions.

In essence the idea is to represent each potential point (which is a vector) as a unique positive integer. By forming constraints which preclude the occurrence of this unique integer representation, the corresponding potential point is excluded from the feasible region.

On each iteration the potential set is updated with respect to the current tradeoff cut. Constraints are added to force all but one potential point in this set to be infeasible to the IP problem. The one potential point which is allowed to remain feasible is the current iteration point. Solving the IP problem returns the next iteration

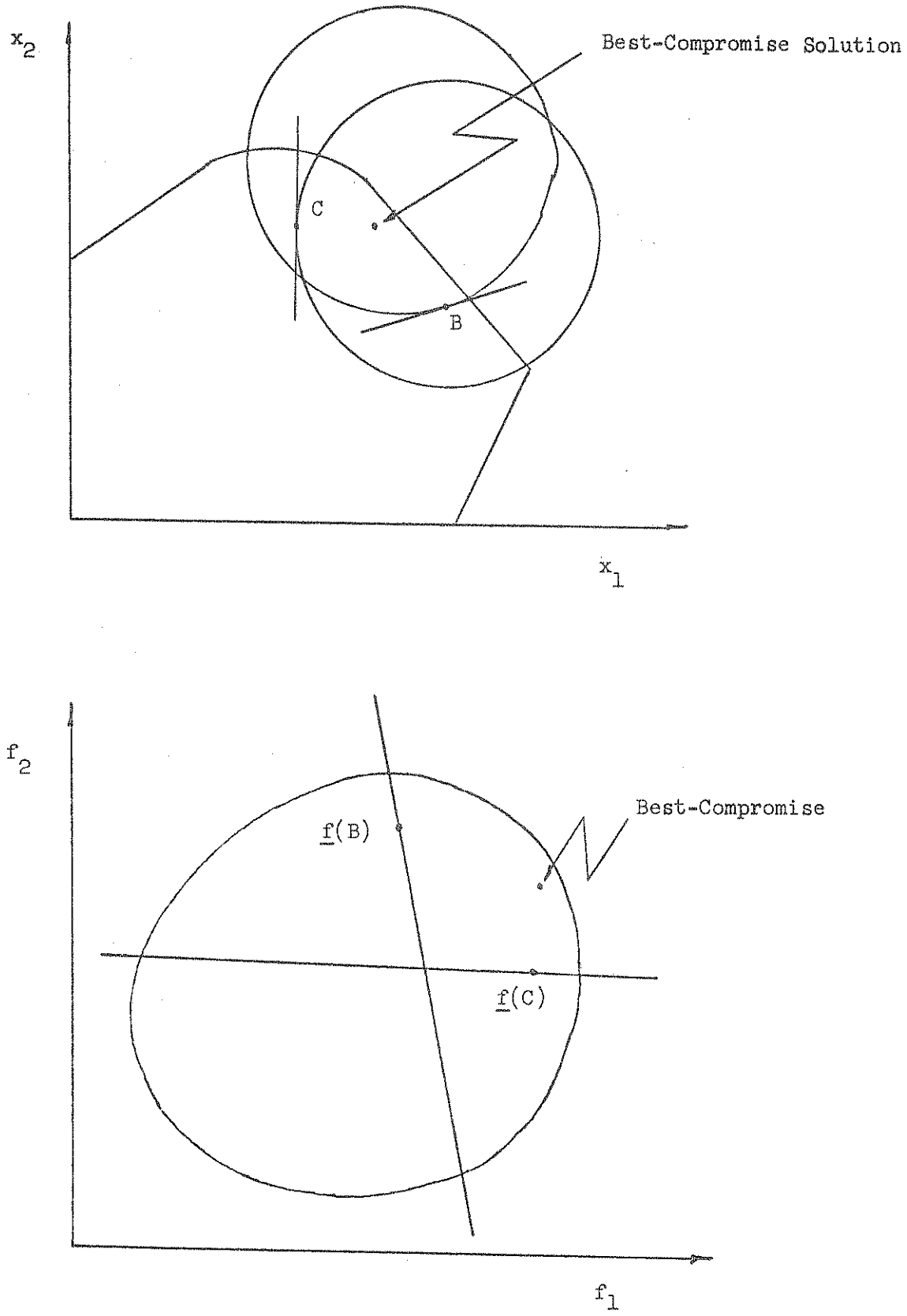


Figure 5.2. Inability to Locate the Best-Compromise Solution

point. This IP problem will always have a solution since the current (integer) iteration points remains feasible.

If the IP problem returns an integer point other than the current iteration point, this new solution becomes the next iteration point. Accordingly, it is designated a potential point and the potential set is once again updated.

On the other hand, if the current iteration point is returned, it indicates that the present solution is locally best. This means that there is no other integer point in the revised feasible region for which the linear approximation of U will give a larger value. At this stage alternate optimal solutions are investigated. If such solutions exist, the process continues; otherwise, the final potential set is complete and the process terminates.

Shortly, it will be shown that this final potential set contains the best-compromise solution. However, there is a possibility that this set might also contain points inferior to this "optimal" solution. The reason for this stems from the algorithm's inability to effectively observe these inferiorities. Knowledge of the decision maker's local tradeoffs is not always enough to insure ending with the best-compromise set.

If further distinction between the final integer points is necessary, it requires additional analysis. The decision maker needs to provide more information about his overall preferences. One means of accomplishing this is to provide the decision maker with a list of these final integer points together with their corresponding functional values. He might then be able to decide which point is best in this list. Another approach,

under a more analytical setting, is a technique called ELECTRE [6, 48]. Based on certain parameters set by the decision maker, it determines those points considered best in terms of a "concordance-discordance" criteria.

5.3. Ancillary Definitions

Although this need for further analysis may arise, it would be careless to quickly deduce from this that the final potential set is of questionable value. In fact, these final potential points satisfy certain properties which make this final set very useful and important. Moreover, one can readily recognize when the best-compromise set and this final potential set are identical. The definitions and propositions which follow will help to clarify these remarks.

Definition. (Preferred (under U))

A point $\underline{x}^1 \in X$ is preferred (under U) to a point $\underline{x}^2 \in X$ if $U(\underline{x}^1) > U(\underline{x}^2)$.

When $\underline{x}^1$ is preferred to $\underline{x}^2$, it must also be preferred (under U) to $\underline{x}^2$ since U satisfies the nonsatiation property (with respect to the f_i). The converse to this is not necessarily true, however. For instance, if $\underline{x}^1$ and $\underline{x}^2$ are both efficient and $U(\underline{x}^1) > U(\underline{x}^2)$, then $\underline{x}^1$ is preferred (under U) to $\underline{x}^2$; but neither point is preferred to the other.

Definition. (Tradeoff Noninferior)

Any two points $\underline{x}^1$ and $\underline{x}^2$ in X are tradeoff noninferior to each other if $\underline{x}^1$ satisfies the decision maker's tradeoff cut at $\underline{x}^2$ and vice-versa.

Points B and C in Figure 5.2 illustrate two points being tradeoff noninferior to each other. The feasibility of these two points is unaffected by the tradeoff cuts at $\underline{f}(B)$ and $\underline{f}(C)$. One should understand that being tradeoff noninferior does not necessarily mean that the two points are efficient. The property only addresses the feasibility of these points in relationship to the two tradeoff cuts.

Definition. (Plausible Set)

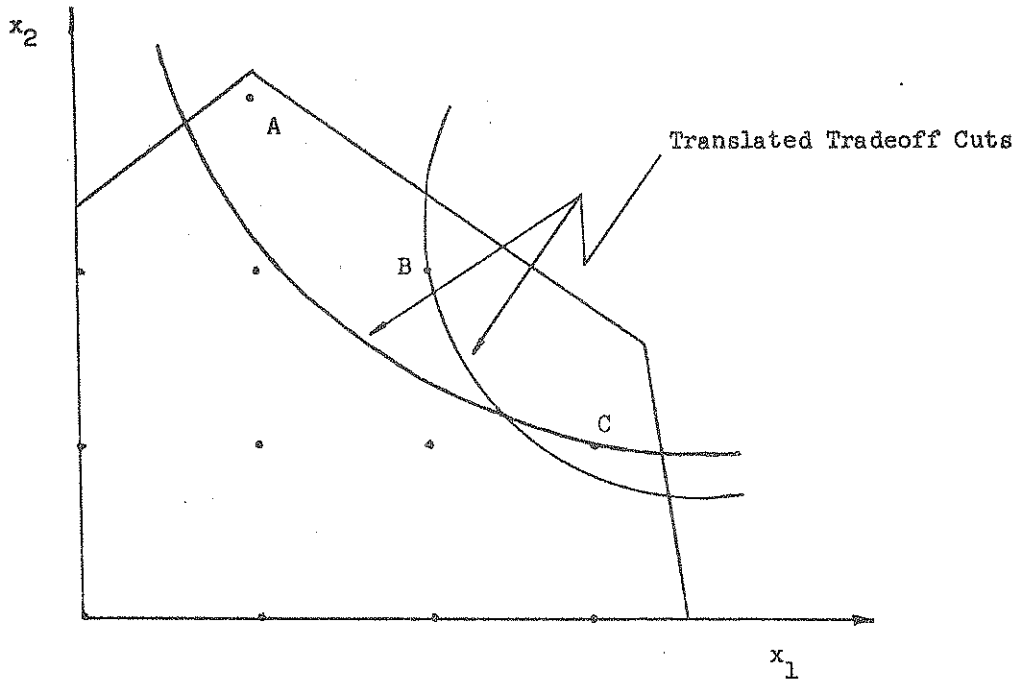
A set $\Omega \subset X_I$ is said to be plausible provided the following two conditions are met:

1. Internal Stability
   a. All points in Ω are efficient.
   b. Any two points in Ω are tradeoff noninferior to each other.
2. External Stability
   a. For any point $\underline{x} \in X_I - \Omega$, there exists at least one point $\underline{x}^* \in \Omega$ which is preferred (under U) to $\underline{x}$.

Definition. (Plausible Points)

The elements of Ω are termed plausible points.

The set {B,C} in Figure 5.3 is a plausible set. Notice that point B is actually preferred (under U) to point C, yet they are both members of the plausible set. This is not in violation of the definition, for elements of Ω may be preferred (under U) to one another. At the same time, being efficient points, no one point is preferred to any other.



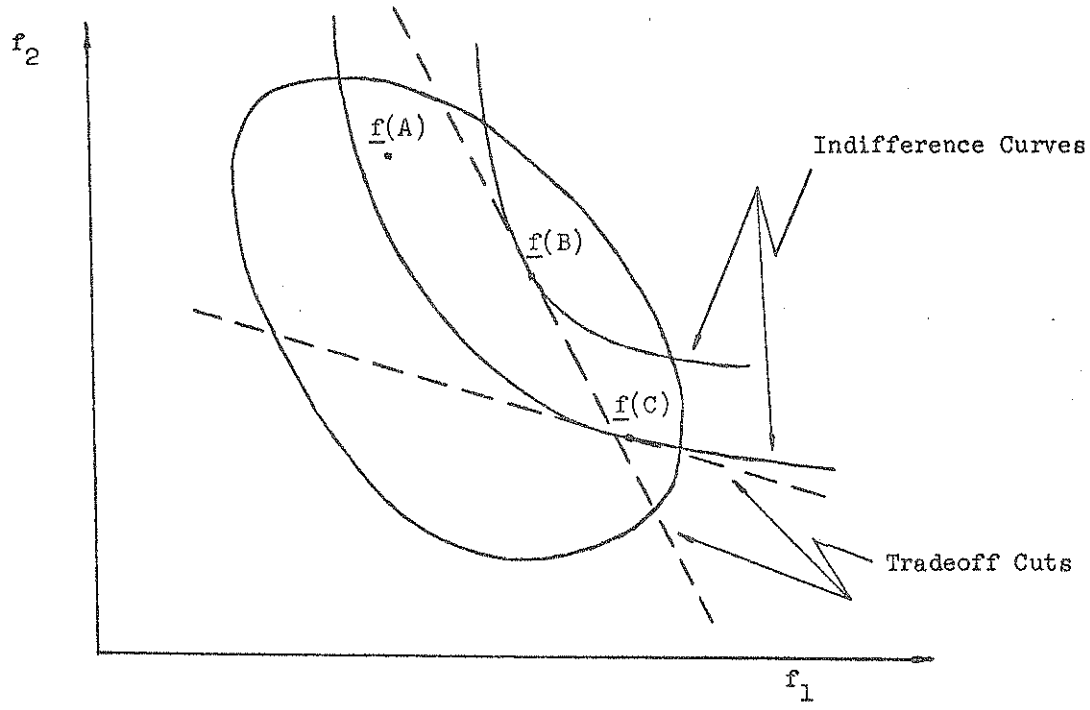


Figure 5.3. The Plausible Set $\Omega = \{ B,C \}$

Point A, outside of Ω, is actually preferred (under U) to point C, a plausible point. On the other hand, point B in Ω is preferred (under U) to point A. Thus, there is at least one element in Ω which is preferred (under U) to point A.

From this example one can understand the meaning behind a set being "nearly" a best-compromise set. The set comprised of point B is the best-compromise set. Ω contains not only point B but also point C. The tangent to U at $\underline{f}(B)$ allows for the inclusion of $\underline{f}(C)$ and, thus, point C. No further reduction in Ω can be accomplished without additional information from the decision maker. Functional tradeoffs at $\underline{f}(B)$ can only produce a linear approximation of U, and this approximation is unable to recognize the inferiority of point C.

The reason for the interest in plausible sets is that the Tradeoff Cutting Plane algorithm for discrete optimization terminates when only plausible points remain in the reduced feasible region. Before proving this fact, it is necessary to state the algorithm.

5.4. Discrete Version of the TCP Algorithm

The TCP algorithm for discrete multiple objective optimization is as follows:

Algorithm.

Step 1. Start with a feasible integer solution $\underline{x}^0$.
Set k=0.

Step 2. Evaluate the functions f_j, $j=1,2,\ldots,r$, at the current iteration point $\underline{x}^k$ (i.e., $f_j(\underline{x}^k)$, $j=1,2,\ldots,r$).

Step 3. Establish tradeoffs at $\underline{f}(\underline{x}^k) = (f_1(\underline{x}^k), \ldots, f_r(\underline{x}^k))$. Let $\underline{t}^k$ stand for the vector whose components are the tradeoffs established at $\underline{f}(\underline{x}^k)$.

Step 4. Form the new tradeoff cut

$$g_{m_k}(\underline{x}) = \sum_{i=1}^{r} t_i^k [\, f_i(\underline{x}^k) - f_i(\underline{x}) \,] \leq 0 .$$

Step 5. Update the set of potential points. That is, delete from the potential set any potential points which fail to satisfy the new tradeoff cut.

Step 6. Solve the following integer programming problem to obtain $\underline{x}^*$.

$$\max \quad \sum_{i=1}^{r} t_i^k \nabla_{\underline{x}} f_i(\underline{x}^k)^t \cdot \underline{x}$$

$$\text{s.t.} \quad g_i(\underline{x}) \leq 0, \quad i=1,2,\ldots,s_k$$

$$x_j, \quad j=1,2,\ldots,n \quad \text{integer}$$

where $s_k = m_0 + k + \ell_k$ and $g_i(\underline{x}) \leq 0$, $i=1,2,\ldots,s_k$

represent the initial m constraints, k tradeoff cuts, and ℓ_k constraints (at the k^{th} iteration) which are added to preclude all but the current potential point from occurring.

Step 7. If $\underline{x}^* = \underline{x}^k$, go to Step 9. Otherwise, continue.

Step 8. Designate $\underline{x}^*$ as a potential point; set $\underline{x}^{k+1} = \underline{x}^*$, set $k = k+1$ and return to Step 2.

Step 9. Test if there is an alternate optimal solution.

a) If there is, denote it as $\underline{x}^*$ and go to Step 8.

b) If not, stop. The current potential set is plausible.

If the original constraints and objective functions are linear, the IP problem in Step 6 of the algorithm is also linear. Solution procedures are readily available to solve this type of problem [44, 45). On the other hand, if any one of these constraints (i.e., original constraints and tradeoff cuts) is nonlinear, the problem becomes more difficult and one's choice of a solution procedure is more limited. Possible selections in this case include several techniques discussed by Garfinkel and Nemhauser in [20], a dynamic programming approach by Cooper [11, 12], Witzgall's all-integer, parabolic programming method [60], and various cutting plane procedures.

5.5. Finiteness of the Algorithm

On each iteration, the algorithm seeks out integer points which have not already been designated as potential points. Past potential points are never considered again for either constraints are added to exclusively avoid them or they are eliminated by tradeoff cuts.

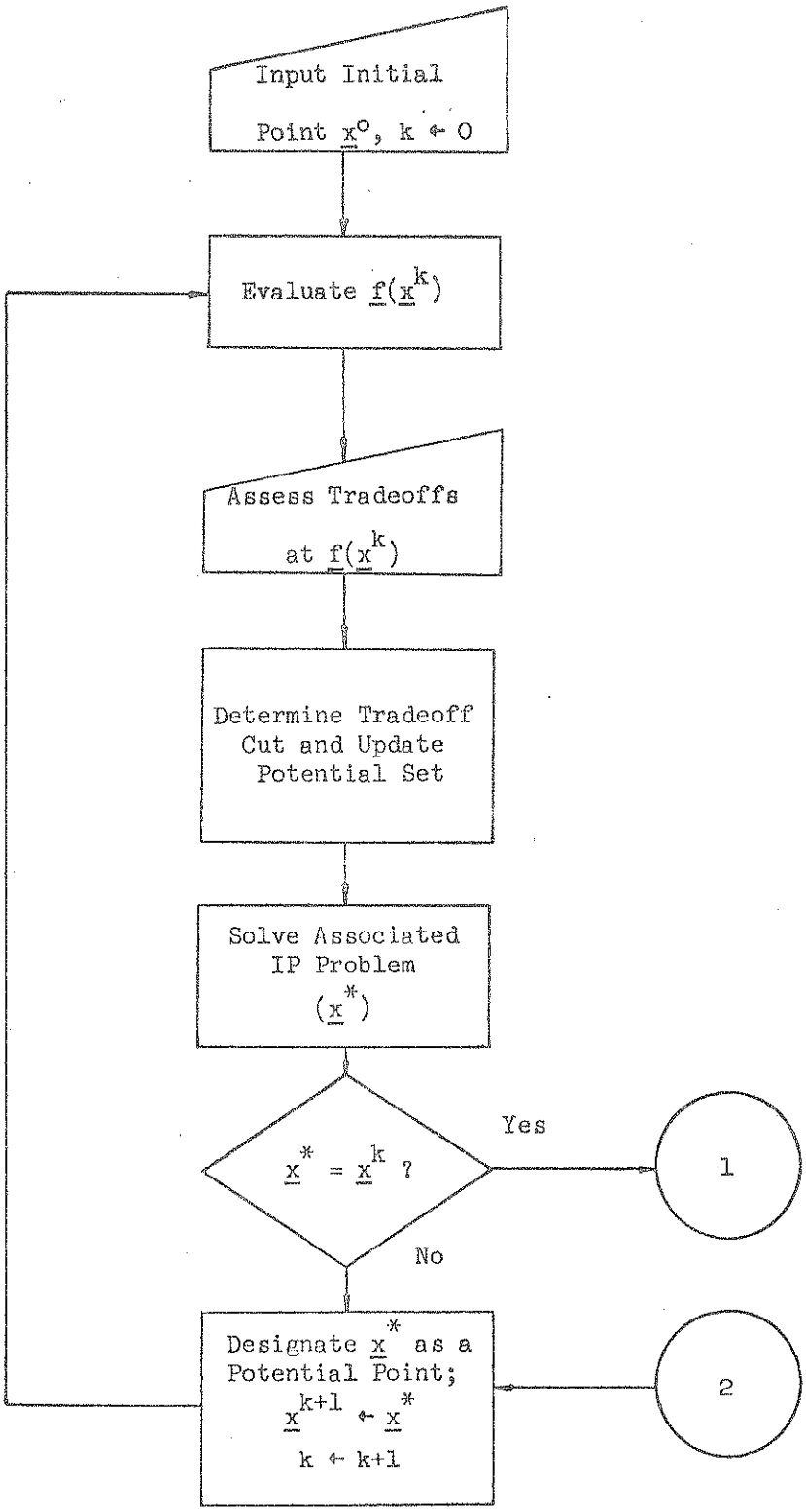


Figure 5.4a. Flowchart of the TCP Algorithm (Discrete Version)

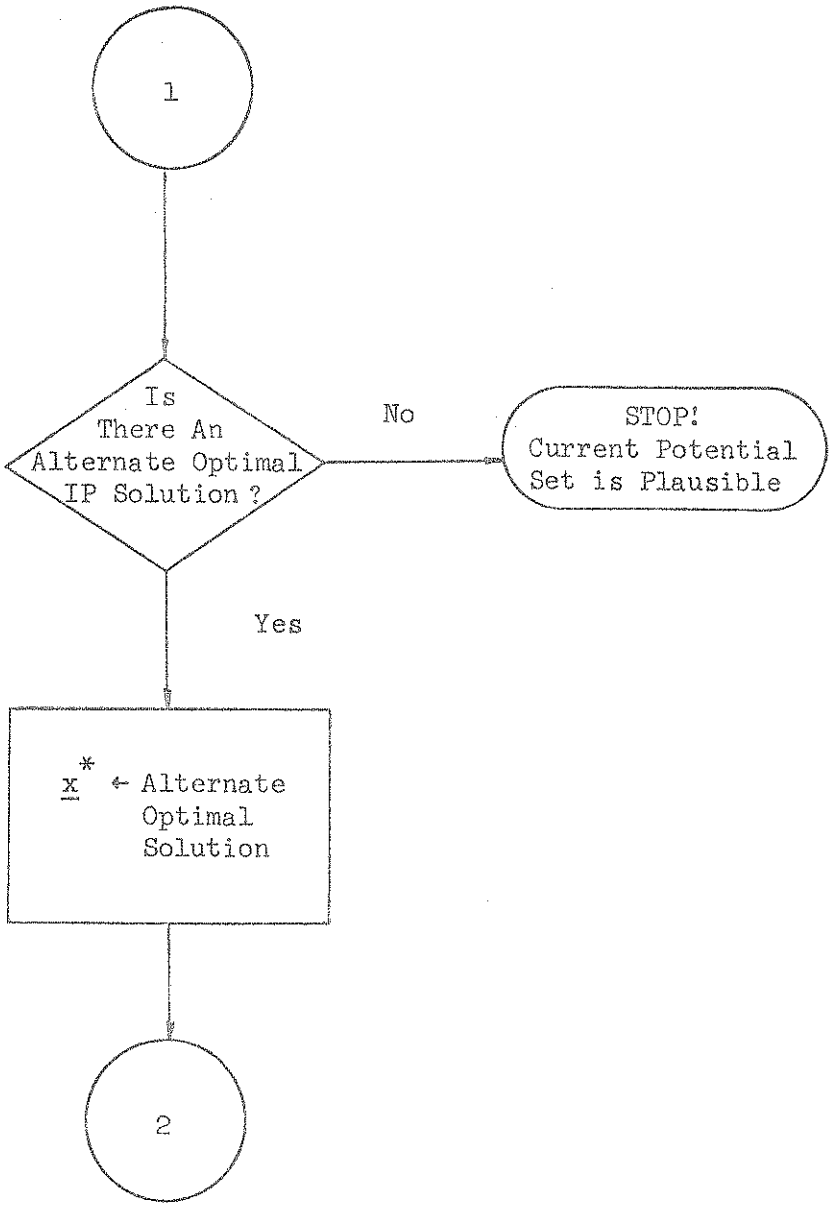


Figure 5.4b. Flowchart of the TCP Algorithm
(Discrete Version)
(Continued)

In the former case they are members of the potential set; in the latter case they simply no longer lie in the updated feasible region. Once the algorithm fails to locate a new potential point, it stops. Since there is only a finite number of points in X_I and no point is ever considered twice, the finiteness of the algorithm is a direct result.

5.6. Results Relating to the Final Potential Set

Let X_f be the final feasible set defined by the algorithm (i.e., the original feasible decision space X_I intersected with each of the translated tradeoff cuts). Let X_p be defined as the potential set upon termination of the algorithm.

Lemma 5.1

$X_p = X_f$.

Proof.

Case 1: $X_P \subset X_F$

By construction any $\underline{x} \in X_P$ must be feasible and satisfy all the accumulated tradeoff cuts. This being the case, $\underline{x} \in X_F$. Hence, $X_P \subset X_F$.

Case 2: $X_F \subset X_P$

Let $\underline{y} \in X_F$. In addition, assume $\underline{y} \notin X_P$. Let $\underline{x}^k$ be the final iteration point of the algorithm. Since $\underline{y}$ is not a potential point and $\underline{x}^k$ is returned at this final stage, the value of the objective function (in Step 6 of the algorithm) at $\underline{x}^k$ exceeds the value at $\underline{y}$. That is,

$$\sum_{i=1}^{r} t_i^k \nabla_{\underline{x}} f_i(\underline{x}^k)^t \cdot \underline{y} < \sum_{i=1}^{r} t_i^k \nabla_{\underline{x}} f_i(\underline{x}^k)^t \cdot \underline{x}^k$$

$$\Rightarrow \alpha \nabla_{\underline{x}} U(\underline{x}^k)^t \cdot \underline{y} < \alpha \nabla_{\underline{x}} U(\underline{x}^k)^t \cdot \underline{x}^k \quad . \tag{5.1}$$

Both $\underline{x}^k$ and $\underline{y}$ being elements of the convex set X_F implies that

$$\underline{x}^k + \lambda(\underline{y} - \underline{x}^k) \in X_F \quad \text{for} \quad 0 \leq \lambda \leq 1 .$$

Therefore, $\underline{y} - \underline{x}^k$ is a feasible direction at $\underline{x}^k$. This being a feasible direction implies that

$$\nabla_{\underline{x}} g_{m_k}(\underline{x}^k)^t \cdot (\underline{y} - \underline{x}^k) \leq 0$$

where $g_{m_k}(\underline{x}^k)$ is the tradeoff cut at $\underline{x}^k$. By substitution,

$$\left[\nabla_{\underline{x}=\underline{x}^k} \left(\sum_{i=1}^{r} t_i^k [f_i(\underline{x}^k) - f_i(\underline{x})] \right) \right]^t \cdot (\underline{y} - \underline{x}^k) \leq 0$$

$$\Rightarrow \left[\nabla_{\underline{x}=\underline{x}^k} \left(\sum_{i=1}^{r} t_i^k f_i(\underline{x}) \right) \right]^t \cdot (\underline{y} - \underline{x}^k) \geq 0$$

$$\Rightarrow \alpha \nabla_{\underline{x}} U(\underline{x}^k)^t \cdot (\underline{y} - \underline{x}^k) \geq 0 \qquad \left(\text{where } \alpha = 1/\left(\partial U / \partial f_1 \right) \Big|_{\underline{f} = \underline{f}(\underline{x}^k)} \right)$$

$$\Rightarrow \alpha \nabla_{\underline{x}} U(\underline{x}^k)^t \cdot \underline{y} \geq \alpha \nabla_{\underline{x}} U(\underline{x}^k)^t \cdot \underline{x}^k .$$

But this is a contradiction to (5.1). Hence, $\underline{y}$ must be a potential point. Since $\underline{y}$ was arbitrarily selected in X_F, $X_F \subset X_P$.

Q.E.D.

Lemma 5.1 means that when the algorithm terminates every point in the final feasible region X_f has been screened as a potential point. As a result, the final potential set and X_f are actually identical.

Considering now those feasible points outside of X_f, Lemma 5.2 investigates their relationship to this final set.

Lemma 5.2

For any feasible integer point not in X_p, there exists at least one point in X_p which is preferred (under U) to it.

Proof.

The initial feasible set X_I is assumed nonempty. Furthermore, the algorithm's final iteration point is integer and feasible. Thus, there always exists at least one point in X_p.

Let $\underline{y}$ be any feasible integer point which is not a member of X_p (i.e., $\underline{y} \in X_I - X_p$). Since it is not in X_p, it must violate at least one tradeoff cut. Say $\underline{y}$ violates the tradeoff cut at $\underline{x}^k$. Then,

$$\sum_{i=1}^{r} t_i^k \left[f_i(\underline{x}^k) - f_i(\underline{y}) \right] > 0.$$

As the result of Lemma 1.2,

$$U(\underline{y}) < U(\underline{x}^k).$$

Therefore, $\underline{x}^k$ is preferred (under U) to $\underline{y}$. Thus, there exists an integer point (viz., $\underline{x}^k$) which is preferred (under U) to $\underline{y}$.

If $\underline{x}^k \in X_p$, then a point in X_p has been found which is preferred (under U) to $\underline{y}$.

If $\underline{x}^k \in X_I - X_p$, consider the following. Any iteration point in $X_I - X_p$ must violate at least one tradeoff cut. Therefore, since the algorithm is finite and transitivity holds, there exists at least one iteration point in X_p which is preferred (under U) to $\underline{x}^k \in X_I - X_p$. By transitivity and the fact that $\underline{x}^k$ is preferred (under U) to $\underline{y}$, there must exist a point (viz., an iteration point) in X_p which is preferred (under U) to $\underline{y}$.

Q.E.D.

Lemma 5.2 opens the possibility of excluding some efficient points from X_p. A point in X_p can never be preferred to an efficient point, but it can be preferred (under U). Thus, requiring preference according to U enables further refinement of the final set.

Lemma 5.3 establishes the type of point found in X_p and states what the relationship is between these potential points.

Lemma 5.3

All points in X_p are:

(i) efficient and

(ii) tradeoff noninferior to each other.

Proof.

Since each potential point also serves as an iteration point, any two points in X_p must be tradeoff noninferior to each other.

A two-part proof is used to show that all points in X_p are efficient. The first part shows that for any point $\underline{x}$ in X_p there is no $\underline{y} \in X_p$ which is preferred to it. The second part shows that the same is true for any $\underline{y} \in X_I - X_p$.

Part I.

Let $\underline{f}(\underline{x}) = (f_1(\underline{x}), f_2(\underline{x}), \ldots, f_r(\underline{x}))$ and, correspondingly, assume $\underline{t} = (t_1, t_2, \ldots, t_r)$ to be the decision maker's tradeoff vector at $\underline{y}$. Recall that $t_i > 0$ for $i=1,2,\ldots,r$. Also, let $\underline{x}$ and $\underline{y}$ be any two potential points in the final set X_p. Since $X_p \subset X_I$, $\underline{x}$ and $\underline{y}$ are both feasible points. If $f_i(\underline{x}) = f_i(\underline{y})$ for each $i=1,2,\ldots,r$, then clearly neither point is preferred to the other. Without loss of generality, assume $f_1(\underline{y}) > f_1(\underline{x})$. Because $\underline{x} \in X_p$, it must satisfy the tradeoff cut at $\underline{y}$. That is,

$$\sum_{i=1}^{r} t_i[f_i(\underline{y}) - f_i(\underline{x})] \leq 0.$$

It follows from this that

$$t_1 f_1(\underline{x}) + \sum_{j=2}^{r} t_j f_j(\underline{x}) \geq t_1 f_1(\underline{y}) + \sum_{j=2}^{r} t_j f_j(\underline{y})$$

Since $f_1(\underline{y}) > f_1(\underline{x})$,

$$t_1 f_1(\underline{y}) + \sum_{j=2}^{r} t_j f_j(\underline{x}) > t_1 f_1(\underline{y}) + \sum_{j=2}^{r} t_j f_j(\underline{y}).$$

Hence, $\sum_{j=2}^{r} t_j f_j(\underline{x}) > \sum_{j=2}^{r} t_j f_j(\underline{y})$.

Since $\underline{t} > 0$, $\exists\, i$, $2 \leq i \leq r$, for which

$$f_i(\underline{x}) > f_i(\underline{y}).$$

This implies that there is a decrease in at least one of the objective functions other than f_1. As a result, since $\underline{x}$ and $\underline{y}$ were arbitrarily selected in X_p, no point in X_p is preferred to any other point in X_p.

Part II.

Now let $\underline{y}$ be any integer point in $X_I - X_p$. Furthermore, assume $\underline{y}$ is preferred to $\underline{x} \in X_p$. That is,

$$f_i(\underline{y}) \geq f_i(\underline{x}) \qquad \text{for } i=1,2,\ldots,r$$

and

$$f_j(\underline{y}) > f_j(\underline{x}) \qquad \text{for some } j=1,2,\ldots,r.$$

Since $\underline{y} \in X_I - X_p$, it must violate a tradeoff cut at some iteration point of the algorithm. Call this point $\underline{w}$. Since $\underline{x} \in X_p$, it must satisfy this same tradeoff cut. That is,

$$\sum_{i=1}^{r} t_i[f_i(\underline{w}) - f_i(\underline{x})] \leq 0 .$$

From the fact that $\underline{y}$ is preferred to $\underline{x}$,

$$\sum_{i=1}^{r} t_i[f_i(\underline{x}) - f_i(\underline{y})] < 0 ,$$

where t_i is the i^{th} component of the tradeoff vector at $\underline{w}$.

Combining these two sums,

$$\sum_{i=1}^{r} t_i[f_i(\underline{x}) - f_i(\underline{y})] + \sum_{i=1}^{r} t_i[f_i(\underline{w}) - f_i(\underline{x})] < 0$$

$$\Rightarrow \quad \sum_{i=1}^{r} t_i[f_i(\underline{w}) - f_i(\underline{y})] < 0 .$$

This, however, contradicts the fact that $\underline{y}$ violates the tradeoff cut at $\underline{w}$. Therefore, $\underline{y}$ cannot be preferred to $\underline{x}$. Thus, there does not exist an integer point in $X_I - X_p$ which is preferred to $\underline{x}$.

From Parts I and II, one must conclude that $\underline{x}$ is an efficient point. Since $\underline{x}$ was arbitrarily selected in X_p, all points in X_p are efficient.

Q.E.D.

Having established the internal and external stability of X_p, the next result is an immediate consequence of Lemmas 5.1, 5.2, and 5.3.

Theorem 5.1

X_p is a plausible set.

As discussed earlier the best-compromise set and the plausible set are very similar. It is possible, though, for the plausible set to contain points which are preferred (under U) to one another. For this reason, X_p will not, in general, be identical to the best-compromise set. Nevertheless, Theorem 5.2 proves that the best-compromise set must be a subset of X_p. The accompanying corollary shows when the two sets are necessarily the same.

Theorem 5.2

The best-compromise set is a subset of the plausible set X_p.

Proof.

Assume there is an integer point $\underline{y}$ in the best-compromise set which is not a member of the plausible set X_p. By Lemma 5.2 there exists an integer point $\underline{x}$ in X_p which is preferred (under U) to $\underline{y}$. But this contradicts the fact that $\underline{y}$ is a best-compromise solution. Therefore, $\underline{y}$ must also be a member of X_p. Hence, the best-compromise set is contained in the plausible set.

Q.E.D.

Corollary 5.2.1

If there is only one element in the final plausible set X_p, then the best-compromise set and the plausible set are identical.

Proof.

From Lemma 5.2 the unique element of X_p is preferred (under U) to any other point in X_I. Consequently, it is the best-compromise solution. In other words, the plausible set is contained in the best-compromise set.

According to Theorem 5.2, the best-compromise set is a subset of the plausible set. Hence, the two sets are identical.

Q.E.D.

The plausible set may be dependent upon which starting point is used to initiate the algorithm. For instance, if one starts at the tradeoff point $\underline{f}(C)$ in Figure 5.5, then $X_p = \{A,C\}$; whereas, by starting at $\underline{f}(B)$, $X_p = \{C\}$. In either case point C, the best-compromise solution, is an element of X_p.

Knowledge of this dependency may be helpful if further analysis of the plausible set is requested. By initiating the algorithm from various starting points and taking the intersection of the resulting plausible sets, further refinement of this final set might be possible. Take note that this intersection is again a plausible set and still must contain the best-compromise solution.

The above argument assumes that the decision maker remains consistent with his preference function over different starting points. As observed in the application presented in Chapter IV, this may not always be the case in practice. Consequently, the intersection of the final plausible sets might turn out to be empty. This would immediately signal that the decision maker is inconsistent and should reconsider his tradeoff assessments.

5.7. Modifications

As it stands, the discrete version of the TCP algorithm might have a tendency to "zig-zag" across the decision space from one boundary to the next. By adding distance restrictions or by employing artificial constraints, one is able to improve the placement of the tradeoff cuts and, thereby, dampen this effect. This section discusses two such approaches to this problem.

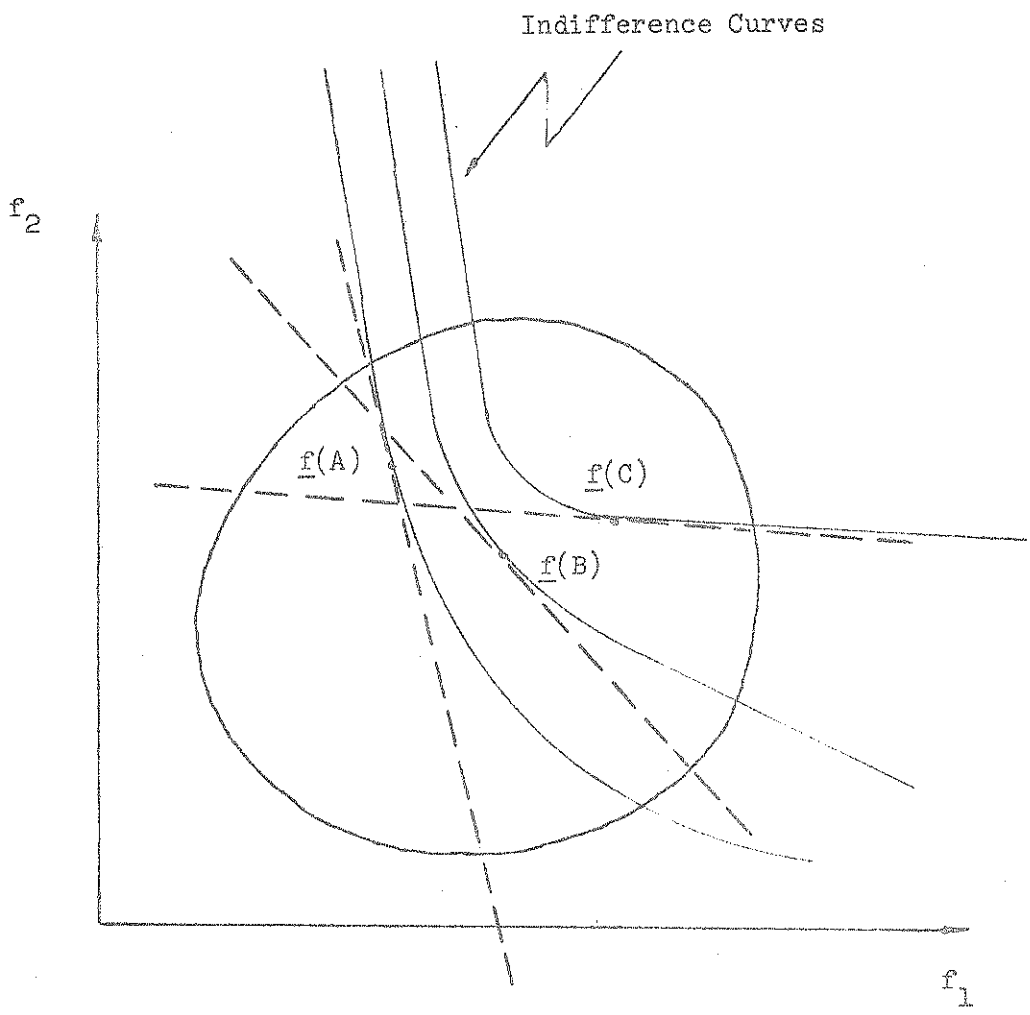


Figure 5.5. Variation in the Final Potential Set

The first of these is an artificial constraint technique which uses what is called a pseudocut. The idea is to solve for $\underline{x}^t = \underline{x}^*$ in Step 6 of the algorithm as usual. Then, the midpoint (or some other suitable point) of the line segment joining the current iteration point $\underline{x}^k$ and $\underline{x}^t$ is calculated. Next, a temporary cut (or pseudocut) is formed which passes through this midpoint and purposefully dissects the decision space. By maximizing the negative of the objective function (found in Step 6) subject to the same constraints plus this cut, one is forced away from the opposite boundary and focuses the direction of the search towards the "center" of the decision space. The new integer solution becomes $\underline{x}^*$ and the process continues to Step 7. The existence of a solution is guaranteed since $\underline{x}^t$ is not a member of the potential set.

An example of a pseudocut is shown in Figure 5.6. It is a displaced linear approximation of U in decision space. After the point $\underline{x}^t$ is determined, the midpoint of the line segment joining $\underline{x}^k$ and $\underline{x}^t$ is calculated. The pseudocut is then centered to pass through this midpoint. It restricts the area of feasibility to that half-space containing $\underline{x}^t$. By maximizing $- \nabla_{\underline{x}} U(\underline{x}^k)^t \cdot \underline{x}$ subject to the updated feasible region and this half-space, added importance is given to those integer points located along the pseudocut. Once this IP problem is solved, the pseudocut is dropped.

Numerous variations to this technique exist. They include retaining some past pseudocuts (thereby causing further restrictions on the search region), temporarily displacing the tradeoff cut to pass through this midpoint, and/or limiting the distance one is allowed

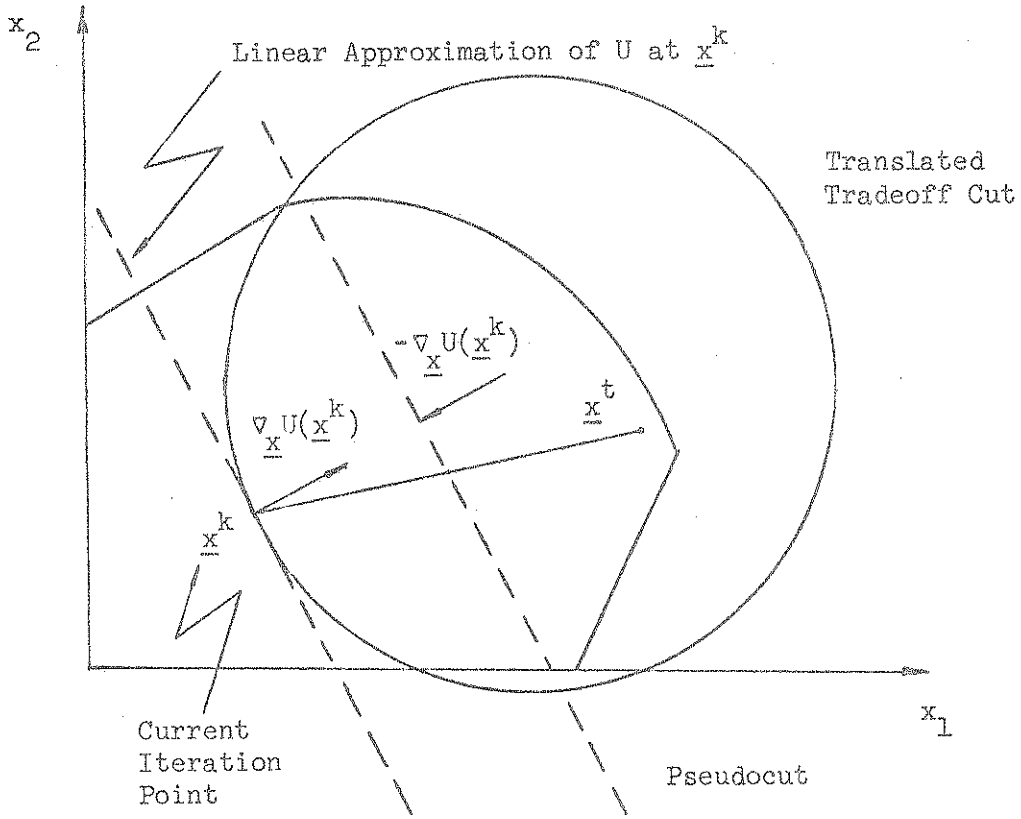


Figure 5.6. Pseudocut Illustration

to search from $\underline{x}^t$. With any of these methods caution is advised for one must remain cognizant of the increased difficulty that may result in trying to solve the associated IP problem.

A second and perhaps superior approach to help prevent zig-zigging is a method which uses a hypercube to restrict movement from the center of the feasible space. It mixes the discrete version of the TCP algorithm (Section 5.4) with the modified method of centers technique explained earlier (Section 3.3). It is termed a hypercube technique and it works as follows. From the current location $\underline{x}^k$, the "center" c of the reduced feasible decision space is determined by first solving (3.6)-(3.9) to obtain $\underline{h}$, next choosing λ^* to be the smallest positive scalar such that

$$d(\underline{x}^k + \lambda^* \underline{h}) = \text{Min}\{\, d(\underline{x}^k + \lambda \underline{h}) \mid \lambda \geq 0 \,\},$$

and then setting $c = \underline{x}^k + \lambda^* \underline{h}$. Although $\underline{x}^k$ has integral coordinates, the center c, in general, does not. Having located the center, the smallest possible hypercube whose vertices have integral coordinates is placed so that it circumscribes this center point. The next iteration point is found by maximizing the linear approximation of U at $\underline{x}^k$ subject to the restrictions imposed by this hypercube and the reduced feasible region. (Recall that the original feasible region only contains integer points.) The image (under $\underline{f}$) of the optimal solution to this subproblem is the next tradeoff point. Having located this point, a new tradeoff cut is established through interaction with the decision maker and the process begins to repeat itself.

If the IP subproblem is infeasible, the dimensions of the hypercube are increased by one unit in each direction. This revised IP subproblem will either be feasible, in which case the process can continue; or infeasible, in which case the dimensions of the hypercube are once again increased. Since the current point $\underline{x}^k$ will eventually be included in the hypercube's expanding boundaries, this IP subproblem must at some point in time admit feasible solutions.

If $\underline{x}^k$ is returned as the solution, the algorithm continues according to the original discrete version of the algorithm. The process can only terminate from this original version. The reason for this is that it is possible to have the current point returned as the optimal solution under this "nested" hypercube approach without detecting the existence of other feasible solutions in the reduced feasible region. For example, let point A in Figure 5.7 be the current location in decision space. Point C is then calculated to be the center point. The algorithm's first hypercube involves, in this case, the square STUV. Since this subproblem is infeasible, the hypercube is expanded to form the square WXYZ. The solution to this problem is point A, the current point. Therefore, having the current point returned as the optimal solution does not imply that the reduced feasible region has been entirely examined, for point B is a feasible solution and it has not been recognized. Now, by returning to the original discrete version of the algorithm, point B is located on the next iteration and the process terminates.

Attaching this hypercube routine to the original discrete version of the TCP algorithm often serves to improve the types of tradeoff cuts used to reduce the decision space. Admittedly, there are situations where the original discrete TCP version is superior to this hybrid

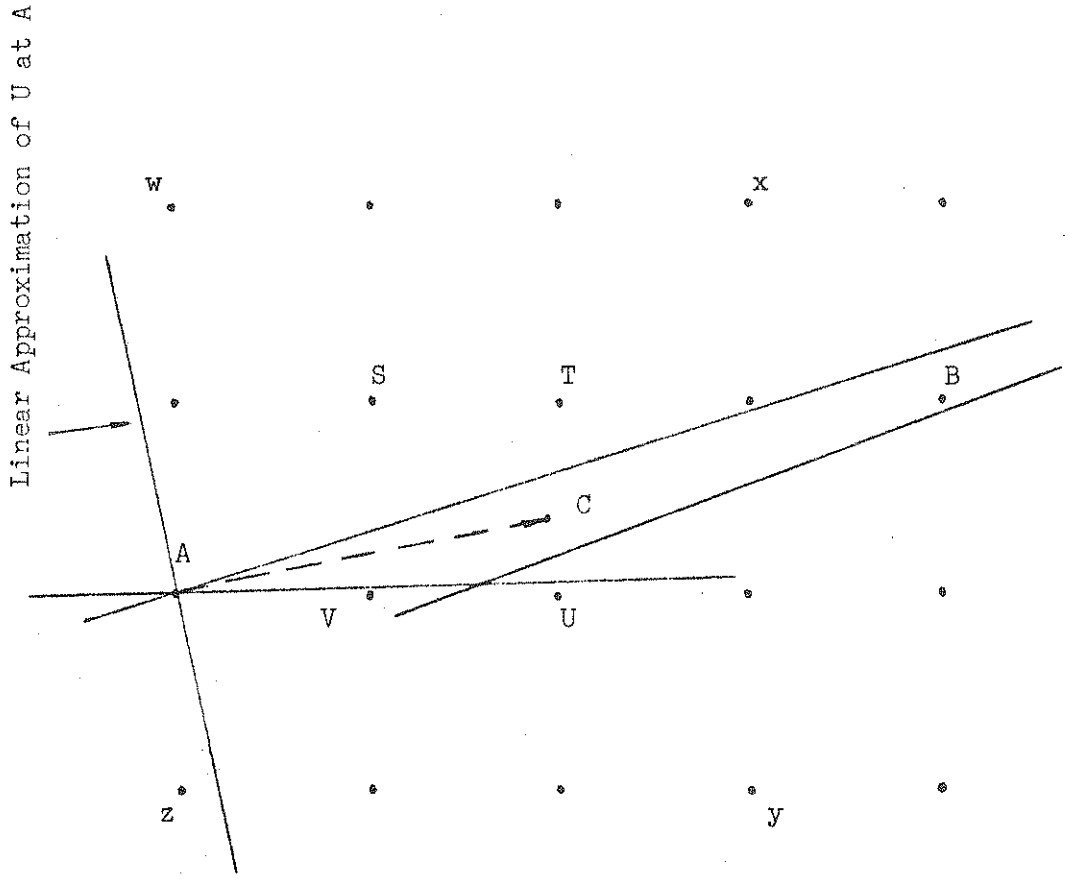


Figure 5.7. Hypercube Caveat

approach. Yet, as a rule, the best cut is usually located near the center of the current feasible region and this hypercube modification is more likely to yield this type of a cut.

In comparing the hypercube and pseudocut routines, it is again not clear as to which of these methods is superior. Depending upon the problem, either routine can be shown to converge in fewer iterations. However, it is the author's contention that the hypercube routine will generally produce a more advantageous tradeoff cut. The reason is again the greater likelihood it has of producing a cut near the center of the current feasible region.

In both of these modification routines (pseudocut and hypercube), the role of the decision maker is unchanged. His only interaction with the algorithm remains to be his assessments of the proportional rates of change among the objectives at various discrete iteration points. Thus, differences among these routines and the basic algorithm are not apparent to the user.

5.8. Examples

In this section the TCP algorithm for discrete optimization is applied to an example problem. Also shown are the pseudocut and nested hypercube routines. The problem used to illustrate these procedures is:

$$\begin{aligned} \text{maximize} \quad & x_1 \\ & x_2 \\ \text{subject to} \quad & 2x_1 + 3x_2 - 12 \leq 0 \\ & -x_1 + 2x_2 - 4 \leq 0 \\ & x_1, x_2 \geq 0 \text{ and integral valued.} \end{aligned}$$

The decision maker's overall preference function is assumed to behave as if

$$U = 20f_1 + 8f_2 - f_1^2 + 2f_1f_2 - 2f_2^2 . \qquad (5.2)$$

Consequently,

$$\partial U/\partial f_1 = 20 - 2x_1 + 2x_2 \quad \text{and} \qquad (5.3)$$

$$\partial U/\partial f_2 = 8 + 2x_1 - 4x_2 . \qquad (5.4)$$

The first procedure shown is the original discrete version of the algorithm. The point (1,2) is used to start the process. The tradeoff vector at this point, found by using (5.3) and (5.4), is (1,1/11). From this, the new tradeoff cut is calculated to be

$$1(1-x_1) + (1/11)(2-x_2) \leq 0$$

$$\Rightarrow \quad -11x_1 - x_2 + 13 \leq 0 . \qquad (5.5)$$

Since only the current point is potential, no updating of the potential set is necessary. Next, the following integer programming problem is solved.

$$\text{maximize} \quad x_1 + \frac{x_2}{11} \qquad (5.6)$$

$$\text{subject to} \quad 2x_1 + 3x_2 - 12 \leq 0 \qquad (5.7)$$

$$-x_1 + 2x_2 - 4 \leq 0 \qquad (5.8)$$

$$-11x_1 - x_2 + 13 \leq 0 \qquad (5.9)$$

$$x_1, x_2 \geq 0 \quad \text{and integral valued.} \qquad (5.10)$$

The solution to this problem is easily identified in Figure 5.8 to be (6,0). Thus, (6,0) is designated as a potential point and it becomes the next iteration point.

Once the objective functions are evaluated at this point, the decision maker again assesses his tradeoffs. His tradeoff vector according to (5.3)-(5.4) is (1,5/2). Accordingly, the second tradeoff cut is determined to be

$$(6-x_1) + (5/2)(0-x_2) \leq 0$$

$$\Rightarrow -2x_1 - 5x_2 + 12 \leq 0 .$$

Since the first potential point (1,2) also satisfies this constraint, no changes are made in the potential set. To preclude the potential point (1,2) from being returned as the optimal solution to the upcoming integer programming problem, the constraints

$$\begin{aligned} x_1 + 10x_2 - My_1 &\leq 20 \\ -x_1 - 10x_2 - My_2 &\leq -22 \\ y_1 + y_2 &= 1 \end{aligned}$$

are added where M is a large positive number and the y_i are zero-one variables. The subsequent IP problem is:

$$\text{maximize} \quad x_1 + (5/2)x_2$$

$$\begin{aligned} \text{subject to} \quad 2x_1 + 3x_2 - 12 &\leq 0 \\ -x_1 + 2x_2 - 4 &\leq 0 \\ -11x_1 - x_2 + 13 &\leq 0 \\ -2x_1 - 5x_2 + 12 &\leq 0 \\ x_1 + 10x_2 - My_1 &\leq 20 \\ -x_1 - 10x_2 - My_2 &\leq -22 \\ y_1 + y_2 &= 1 \end{aligned}$$

$$y_1, y_2 = 0,1; \quad x_1, x_2 \geq 0 \text{ and integral valued.}$$

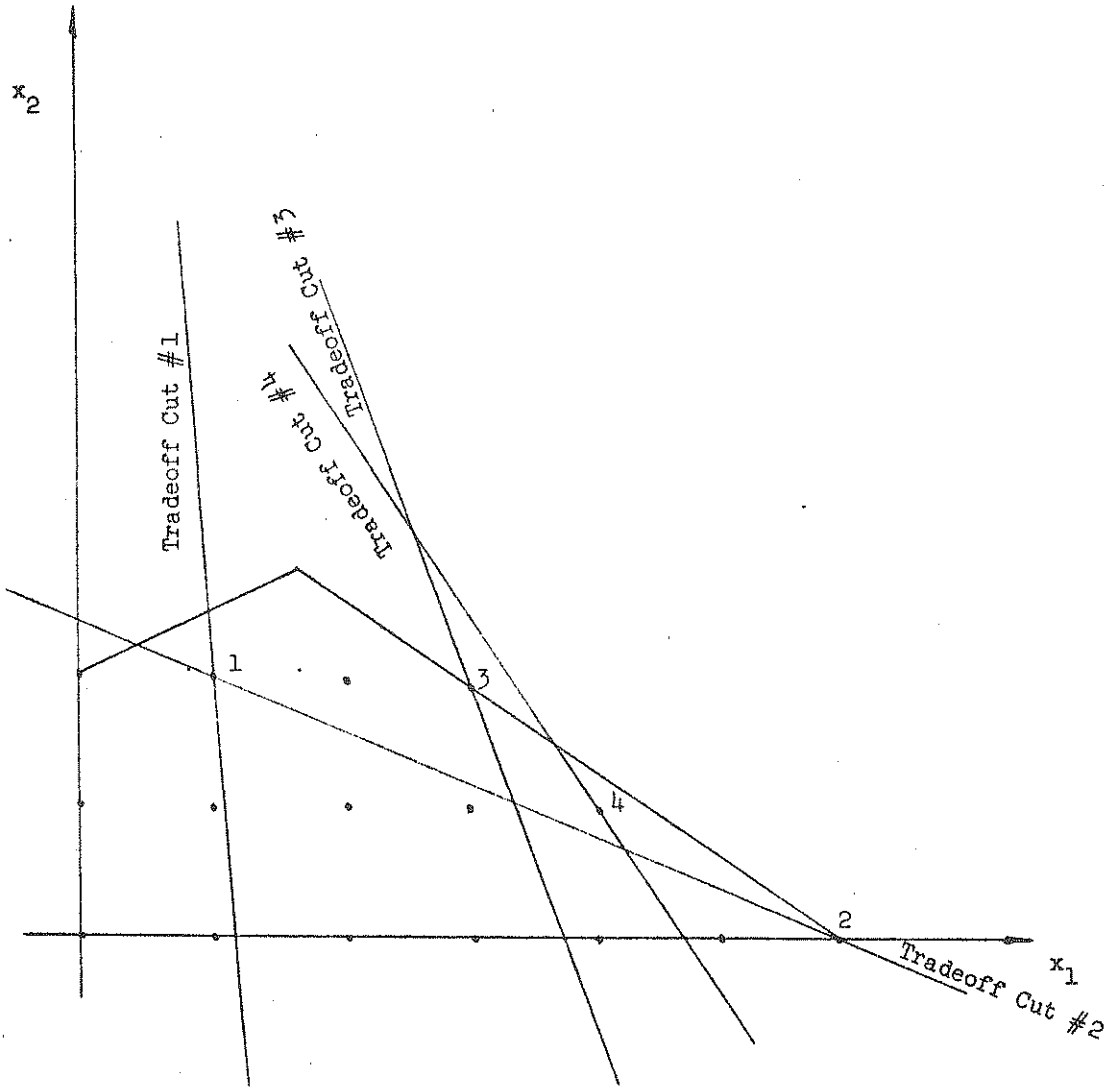


Figure 5.8. Tradeoff Cuts Connected with the Discrete Version of the TCP Algorithm

The optimal solution can again be seen graphically to be (3,2). Its corresponding tradeoff cut translates into

$$-3x_1 - x_2 + 11 \leq 0 .$$

In updating the potential set, the point (1,2) must be eliminated for it fails to satisfy this latest tradeoff cut. The constraints used to circumvent this point, namely

$$\begin{aligned} x_1 + 10x_2 - My_1 &\leq 20 \\ -x_1 - 10x_2 - My_2 &\leq -22 \\ y_1 + y_2 &= 1 \\ y_1, y_2 &= 0, 1 , \end{aligned}$$

are now also discarded. The new potential set contains only (6,0) and (3,2).

On the next iteration the feasible point (4,1) is discovered and its tradeoff cut rejects (3,2) as a potential point. At this stage the algorithm terminates for there are no new feasible points remaining in the reduced feasible region (Figure 5.8). The final potential set, {(4,1),(6,0)}, is plausible. As for it being the best-compromise set, this cannot be determined for there is more than one element in the set.

Using the decision maker's overall preference function (5.2), the best-compromise solution is established to be (6,0). Tradeoffs alone, however, cannot distinguish between (4,1) and (6,0). More information from the decision maker is required before a distinction between these two points can be made.

Had the original problem included, for example, the constraint

$$2x_1 - 9x_2 - 9 \leq 0,$$

then the final potential set would have contained only the point (4,1), since (6,0) fails to satisfy this constraint. This being the case, the final potential set and the best-compromise set would be identical, revealing (4,1) as the best-compromise solution.

Under the pseudocut modification, the sequence of points followed to the final potential set changes slightly. Starting again at (1,2), the tradeoff cut is as before in (5.5) and the same IP problem (5.6)-(5.10) is solved to establish (6,0) as the solution. Here, however, the process does not accept (6,0) as the next iteration point, but instead forms a pseudocut which dissects the line segment between (1,2) and (6,0). The linear approximation of U at (1,2) is displaced to pass through the midpoint of this line segment which is at (7/2,1). The pseudocut restricts the feasible region to include that portion of the space containing (6,0) (Figure 5.9). This cut is determined to be

$$11x_1 + x_2 \geq 79/2.$$

The IP problem is now changed to include this pseudocut (which temporarily replaces the former tradeoff cut) and to maximize the negative of the previous objective function. It reads as follows:

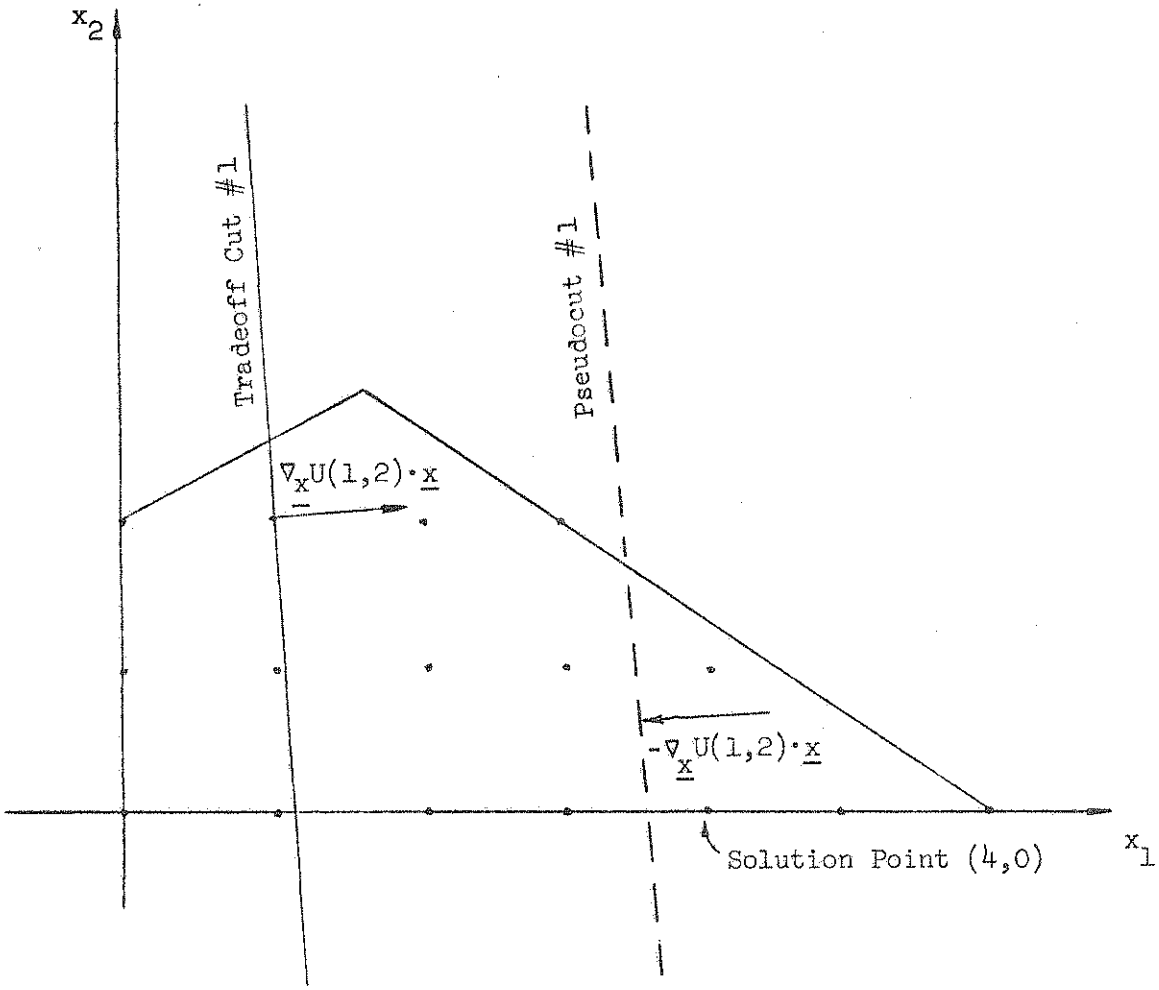


Figure 5.9. First Iteration Pseudocut

$$\text{maximize} \quad -x_1 - \frac{x_2}{11}$$

$$\text{subject to} \quad 2x_1 + 3x_2 - 12 \leq 0$$
$$-x_1 + 2x_2 - 4 \leq 0$$
$$-11x_1 - x_2 + 79/2 \leq 0$$
$$x_1, x_2 \geq 0 \text{ and integral valued.}$$

The optimal solution to this problem, (4,0), becomes the next iteration point. In updating the potential set, one finds that the tradeoff cut at (4,0) eliminates (1,2) as a potential point. The next iteration point under this pseudocut approach turns out to be (4,1). Its tradeoff cut eliminates the former potential point (4,0). On the next iteration, the point (6,0) is found to be feasible. The process then terminates with the final potential set again including (4,1) and (6,0) (Figure 5.10).

With the hypercube approach, the direction to the center of the feasible region is determined by using (3.6)-(3.9). Starting at point (1,2), this direction is calculated to be (1,0). Recall that the tradeoff cut at (1,2) is included as a constraint in determining this direction. By minimizing the maximum distance to all the constraints, the center of this reduced feasible region is determined to be at (2.33,2). The points (2,2) and (3,2) define the smallest "hypercube" which circumscribes this center point and whose vertices have integral coordinates (Figure 5.11).

The next iteration point, (3,2), is determined by using the linear approximation of U at (1,2) and maximizing it subject to the restrictions imposed by the original constraints, the accumulated

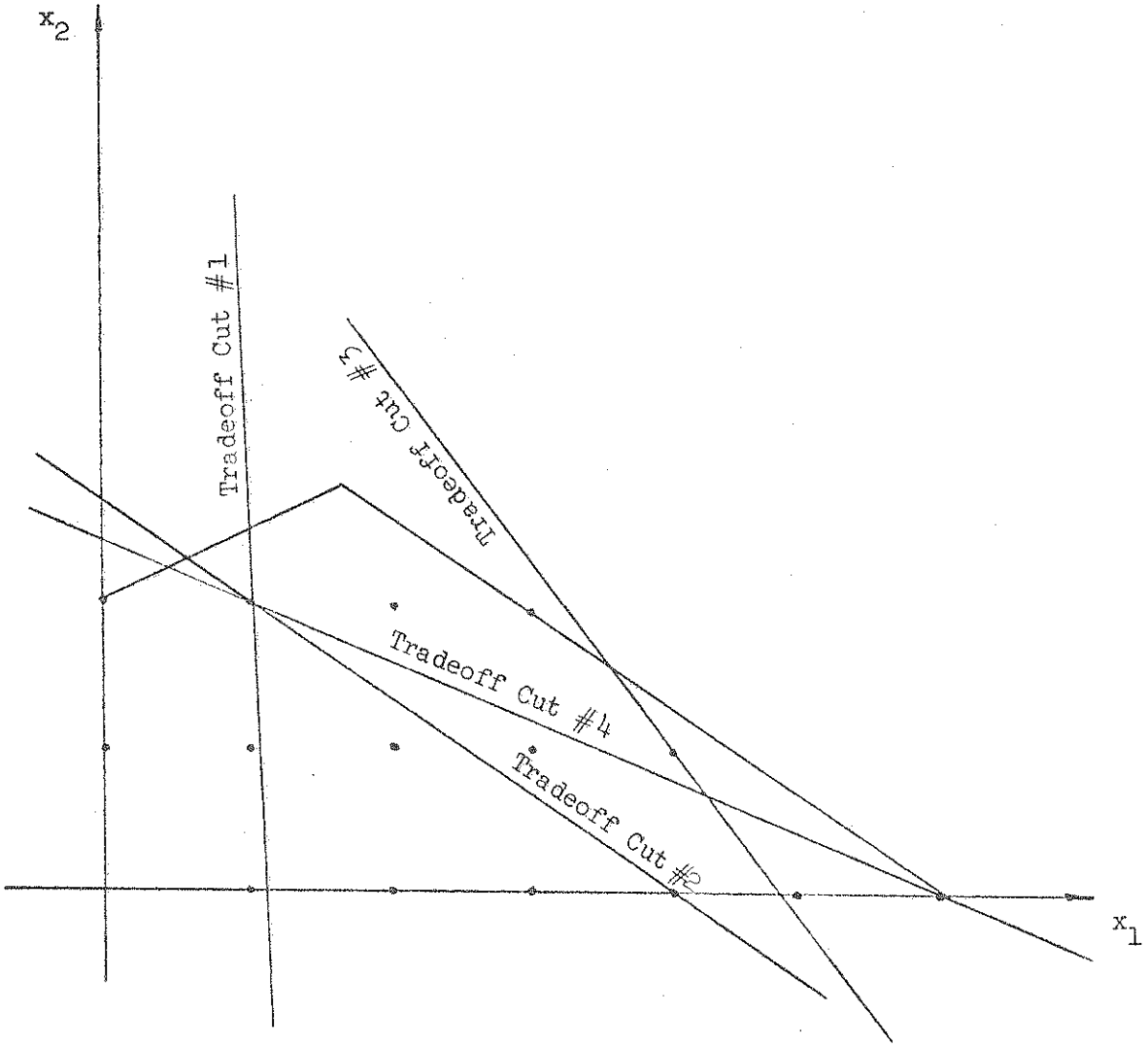


Figure 5.10. Tradeoff Cuts Connected with the Pseudocut Routine

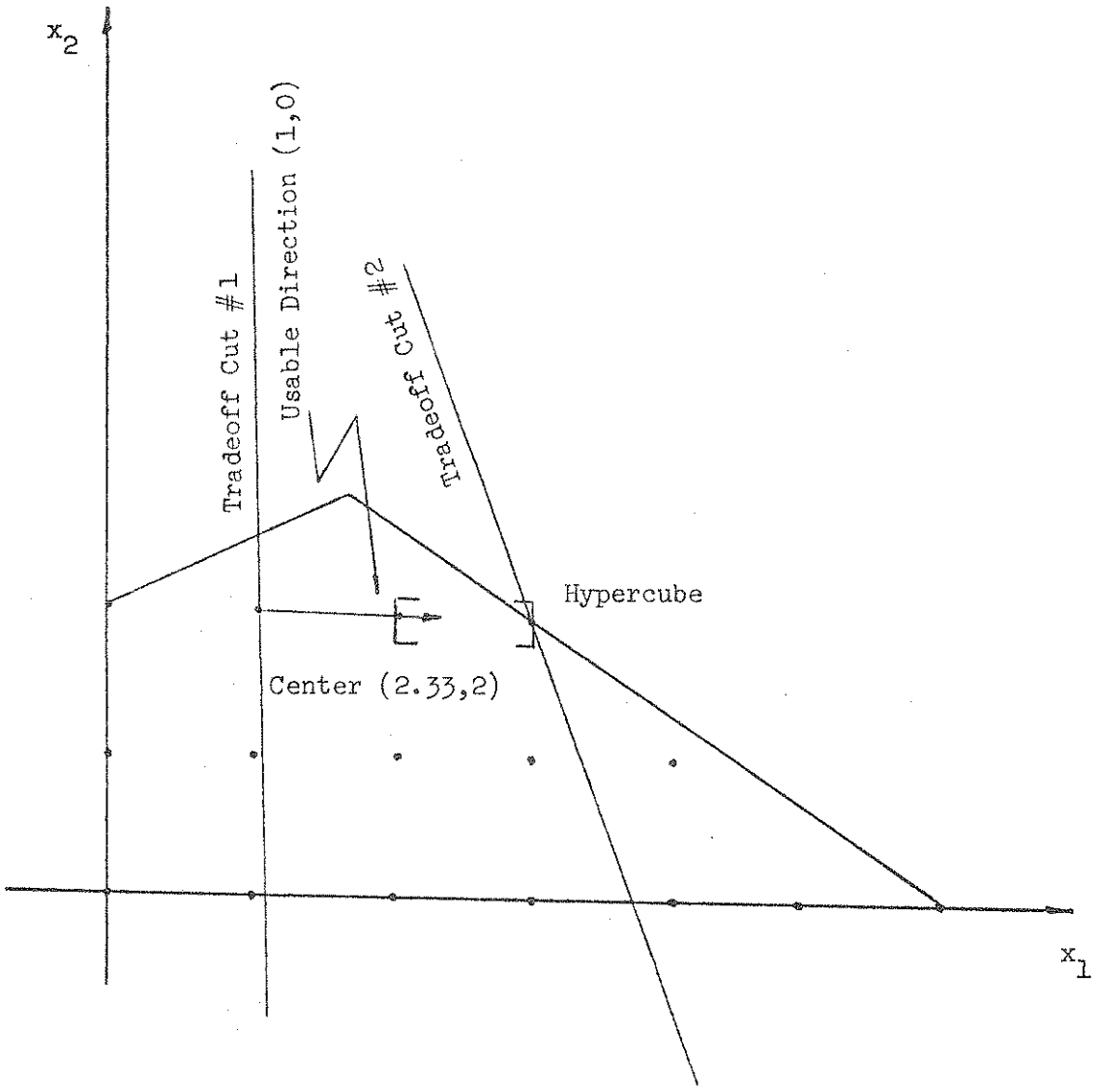


Figure 5.11. First Iteration of the Hypercube Routine

tradeoff cuts, and the boundaries of the hypercube. The image (under f) of (3,2) is the next tradeoff point. The feasible region is then reduced again by introducing the next tradeoff cut,

$$-x_1 - (1/3)x_2 + (11/3) \leq 0 .$$

Solving for the next usable direction determines the direction to the center of this newly revised feasible region. This direction is established to be (0.667,-0,833) and results in the new center point being located at (4.09,0.64). The points (4,0), (4,1), (5,0) and (5,1) define the boundaries of the next "hypercube". As can be seen in Figure 5.12, the point (5,0) becomes the next iteration point.

The next two iterations produce the remaining two feasible points, (4,1) and (6,0). Since their respective tradeoff cuts eliminate all other feasible candidates, the process terminates with the final set once again containing these two points.

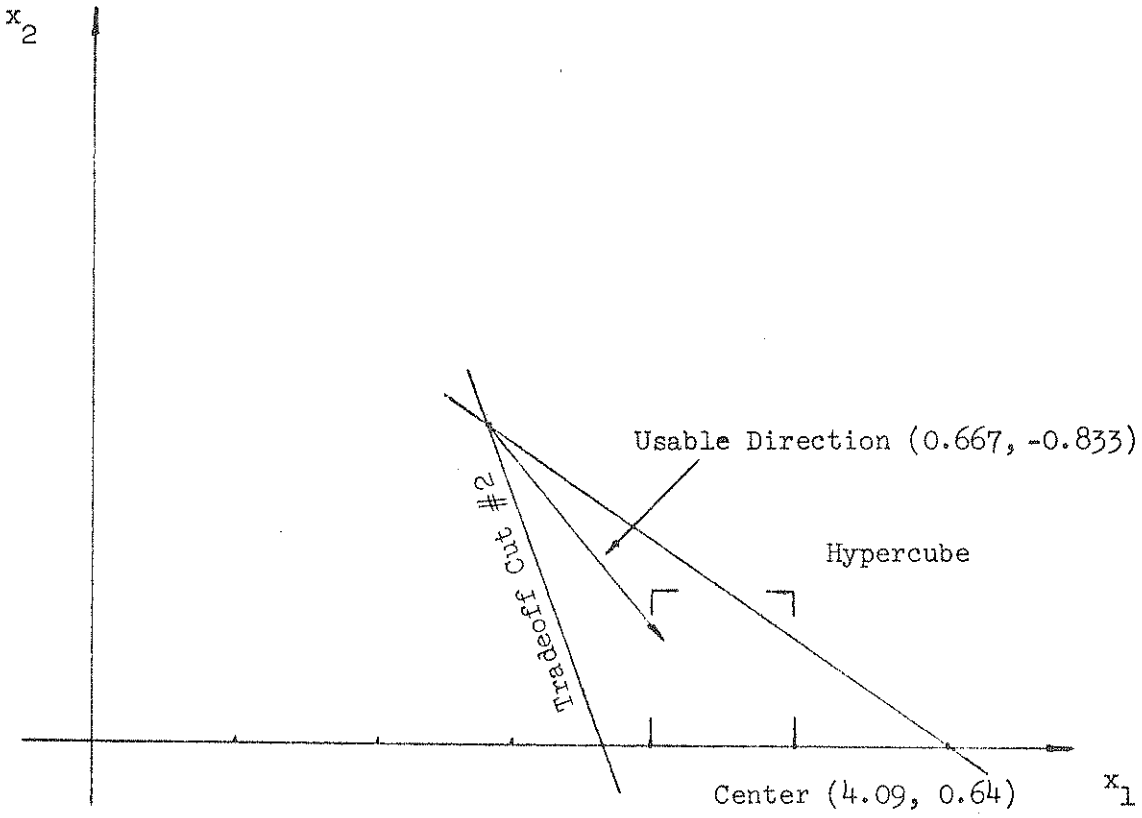


Figure 5.12. Second Iteration of the Hypercube Routine

CHAPTER VI

SUMMARY AND RECOMMENDATIONS

This chapter summarizes the material presented in the previous chapters and discusses its significance to the field of multiple objective decision analysis. Following this, several recommendations for further analysis are briefly noted.

6.1. Overview

As previously discussed in Chapter I, it is becoming increasingly important to develop realistic, user-oriented, mathematical techniques for the resolution of decision problems in which several objectives must be considered simultaneously. More effort is continually being devoted to the development of these solution procedures. Presently, there exist many, widely varied approaches to this type of problem. One such approach, the Geoffrion-Dyer-Feinberg algorithm, has received a great deal of attention in this area. The method answers the complex multiple objective question by allowing interaction between the computer and the decision maker during the solution process. The express purpose for this interaction is to indirectly unify the diverse and often conflicting objectives associated with the problem.

This same approach is adopted with the technique developed in this thesis, the Tradeoff Cutting Plane algorithm. However, contrary

to what is seen in the literature and the GDF algorithm in particular, this technique focuses its attention on the decision space. Through the use of "tradeoff cuts", the convex envelope containing the best-compromise solution is continually refined. The approach concentrates on reducing the feasible region in decision space, rather than improving one's knowledge of the overall preference function. In so doing, the problem is reduced to a series of pairwise decisions between the objective functions. This obviates the need for any type of vector decision on the part of the decision maker and stays reasonably within his capability to supply the necessary information to resolve this type of complex decision problem. Equally noteworthy is the fact that this approach is applicable to both continuous and discrete optimization problems, making it a practical and versatile technique to consider.

6.2. Areas for Future Research

While the Tradeoff Cutting Plane algorithm is a promising and useful approach to the multiple objective problem, there are still difficulties which need to be overcome. Foremost among these is one's tendency to blindly apply multiple objective decision analysis techniques. Hopefully, the TCP algorithm will help to bridge the gap which still exists between the skills needed to effectively employ these types of algorithms and the aptitude possessed by the practitioners who are anxious to benefit through their use. Any multiple objective optimization technique should be applied cautiously, for the success of any of these methods is limited by the competence of those who apply it.

The TCP algorithm is only viewed as another step in the quest to find a universally acceptable means of resolving multiple objective optimization problems. Still further extensions and improvements are envisioned for this algorithm. Challenging areas for future work can be divided into two general categories: (i) algorithmic improvements and (ii) general suggestions. The following list includes recommendations in both of these categories.

Algorithmic Improvements

1. Examine better ways of determining a center point for the reduced feasible region. Currently, the center is located on the basis of a distance measure which is dependent upon the magnitudes of the constraints. Freedom from this dependence would simplify the implementation of the algorithm.

2. Investigate the effect of using different "centers". Presently, the center is defined as the point along the linear program's usable direction ray which maximizes the minimum distance to all the constraints. Possibly a better center point could be obtained by more adeptly incorporating the particular features of the problem.

3. Blend the two versions of the TCP algorithm into a unified procedure which will extend the application of this method to problems involving mixed integer solutions. This would preclude the need for the existence of two separate versions of the algorithm and greatly improve its applicability to real-world decision models.

4. Improve the convergence characteristics of the algorithm in the neighborhood of the best-compromise solution. One approach to this problem is to assume or acquire more information about the overall preference function U. Presently, a linear approximation to U is used throughout the analysis. By using a quadratic approximation to U, for example, the convergence rate should improve. As mentioned above in (2), using different schemes to locate the center point is another relevant consideration. Possibilities here range from devising a distance measure which does not rely on the magnitude of the constraints to running experiments to determine what "relaxation" factor best accommodates this general class of problems. Finally, an examination of the effect of using different reference criteria as one progresses towards the final solution might also prove fruitful.

5. Improve computational efficiency by devising ways of discarding former tradeoff cuts once they no longer border on the current feasible region.

6. Formulate various program termination schemes. Presently, the TCP algorithm terminates whenever there is an error, $M \geq -0.00001$, $|h_i| \leq 0.00001$ for all i, or it is instructed to do so by the decision maker.

7. Improve the type of information given the decision maker as he is progressing towards the best-compromise solution,

helping him to better understand his progress and the particular characteristics of his problem. Suggestions include determining the Euclidean distance between successive iteration points and displaying the distance to each constraint. Also associated with this area is the need to better equip the decision maker with information pertinent to his tradeoff assessments. Devising schemes to indirectly arrive at these tradeoffs (by more closely matching the decision maker's background and understanding) would prove to be very useful.

General Suggestions

1. Develop realistic, accurate and consistent means of assessing tradeoffs. An advancement in this area would result in notable improvements in not only the TCP algorithm, but also many other procedures which rely heavily on the reliability of these tradeoffs.

2. Extend the concepts introduced with the TCP algorithm to the much broader class of problems associated with decision making under uncertainty. In Chapter IV regression approximations to the STORM simulation output were used as a means of handling the stochastic nature of that problem. Yet, this does not, generally speaking, completely resolve the uncertainty question. Considerably more remains to be done in the way of transferring this uncertainty through the analysis to the final solution set.

3. Extend the applications of the TCP method to analyses involving multiple decision makers. Some progress has been made toward this end by the elimination of the difficult vector decisions which previously had to be made along the usable direction, but further advancements need to be made before multiple-member decision processes can be realistically and effectively modeled.

4. Weaken some of the more restrictive assumptions of the problem (e.g., the concavity of the f_i and the convexity of the g_j) while maintaining conditions sufficient to ensure the convergence of the algorithm.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Baumol, W.J., Economic Theory and Operations Analysis, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.

2. Bazaraa, M.S., "An Efficient Cyclic Coordinate Method for Optimizing Penalty Functions", Working Paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1973.

3. Belenson, S.M. and Kapur, K.C., "An Algorithm for Solving Multi-criterion Linear Programming Problems with Examples", Operations Research Quarterly, Vol. 24, No. 1, March 1973, pp. 65-77.

4. Benayoun, R., de Montgolfier, J., Tergny, J., and Laritchev, O., "Linear Programming with Multiple Objective Functions: Step Method (STEM)", Mathematical Programming, Vol. 1, No. 3, December 1971, pp. 366-375.

5. Benayoun, R., Tergny, J., and Keuneman, D., "Mathematical Programming with Multi-objective Functions: A Solution by P.O.P. (Progressive Orientation Procedure)", Metric, Vol. IX, No. 2, June 1970, pp. 279-299.

6. Benayoun, R., Roy, B., and Sussmann, B., "ELECTRE: Une Methode pour Guider le Choix en Presence de Points de Vue Multiples", Note de Travail No. 49, Direction Scientifique SEMA, June 1966.

7. Bitran, G.R., "Linear Multiple Objective Programs with Zero-One Variables", Mathematical Programming, Vol. 13, No. 2, October 1977, pp. 121-139.

8. Bui-Trong-Lieu and Huard, P., "La Methode des Centres dans un Espace Topologique", Numerische Mathematik, Vol. 8, March 1966, pp. 56-67.

9. Charnes, A. and Cooper, W.W., Management Models and Industrial Applications of Linear Programming, John Wiley & Sons, Inc., New York, 1961.

10. Cohon, J.L. and Marks, D.H., "A Review and Evaluation of Multi-objective Programming Techniques", Water Resources Research, Vol. 11, No. 2, April 1975, pp. 208-220.

11. Cooper, L. and Cooper, M.W., "Non-Linear Integer Programming", Comp. & Maths. with Appls., Vol. 1, 1975, pp. 215-222.

12. Cooper, M.W., "An Improved Algorithm for Non-Linear Integer Programming", Technical Report IEOR 77005, Department of Industrial Engineering and Operations Research, Southern Methodist University, February 1977.

13. Cyert, R.M. and March, J.G., A Behavioral Theory of the Firm, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1963.

14. Dendrou, S., "Multilevel Approach to Urban Storm Water Systems Planning", Ph.D. Thesis, Purdue University, West Lafayette, Indiana, December 1977.

15. Dyer, J.S., "A Time Sharing Computer Program for the Solution of the Multiple Criteria Problem", Management Science, Vol. 19, No. 12, August 1973, pp. 1379-1383.

16. Dyer, J.S. and Sarin, R.K., "Multicriteria Decision Making", Encyclopedia of Computer Science and Technology, December 1977.

17. Fishburn, P.C., "Lexicographic Orders, Utilities and Decision Rules: A Survey", Management Science, Vol. 20, 1974, pp. 1442-1471.

18. Fishburn, P.C., The Theory of Social Choice, Princeton University Press, Princeton, New Jersey, 1973.

19. Frank, M. and Wolfe, P., "An Algorithm for Quadratic Programming", Naval Research Logistics Quarterly, Vol. 3, No. 1, March-December 1956, pp. 95-110.

20. Garfinkel, R.S. and Nemhauser, G.L., Integer Programming, John Wiley & Sons, Inc., New York, 1972.

21. Geoffrion, A.M., "Vector Maximal Decomposition Programming", Working Paper No. 164, Western Management Science Institute, UCLA, September 1970.

22. Geoffrion, A.M., Dyer, J.S., and Feinberg, A., "An Interactive Approach for Multi-Criterion Optimization with an Application to the Operation of an Academic Department", Management Science, Vol. 19, No. 4, December 1972, pp. 357-368.

23. Griffith, R.E. and Stewart, R.A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", Management Science, Vol. 7, No. 4, 1961, pp. 379-392.

24. Haimes, Y.Y. and Hall, W.A., "Multiobjectives in Water Resources Systems Analysis: The Surrogate Worth Trade Off Method", Water Resources Research, Vol. 10, No. 4, August 1974, pp. 615-624.

25. Haimes, Y.Y. and Hall, W.A., "The Surrogate Worth Trade-Off Method with Multiple Decision-Makers", In Zeleny, M. (Ed.), Multiple Criteria Decision Making: Kyoto 1975, Springer-Verlag, New York, 1975.

26. Haimes, Y.Y., Hall, W.A., and Freedman, H.T., Multiobjective Optimization in Water Resources Systems (SWT), Elsevier Scientific Publishing Company, Amsterdam, 1975.

27. Himmelblau, D.M., Applied Nonlinear Programming, McGraw-Hill, New York, 1972.

28. Hogan, W.W., "Convergency Results for Some Extensions of the Frank-Wolfe Method", Working Paper No. 169, Western Management Science Institute, UCLA, January 1971.

29. Huard, P., "Programmation Matematique Convexe", Rev. Fr. Inform. Rech. Operation., No. 7, 1968, pp. 43-59.

30. Huard, P., "Resolution of Mathematical Programming with Nonlinear Constraints by the Method of Centres", Nonlinear Programming, Abadie, J. (Ed.), North-Holland Publishing Co., Amsterdam, 1967.

31. Ignizio, J.P., Goal Programming and Extensions, D.C. Heath and Company, Lexington, Massachusetts, 1976.

32. Intriligator, M.D., Mathematical Optimization and Economic Theory, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

33. Juergens, L.J. and Ravindran, A., "A Constraint Method for Generating Multiple (or Next Best) Optimal Solutions to Integer Programming Problems", School of Industrial Engineering, Purdue University, West Lafayette, Indiana, October 1976.

34. Keeney, R.L. and Raiffa, H., Decisions with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, New York, 1976.

35. Klahr, C.N., "Multiple Objectives in Mathematical Programming", Operations Research, Vol. 6, No. 6, November-December 1958.

36. Lee, S.M., Goal Programming for Decision Analysis, Amerbach Publishers, 1972.

37. Lee, S.M., "Interactive and Integer Goal Programming", Management Science, (forthcoming).

38. MacGrimmon, K.R., "An Overview of Multiple Objective Decision Making", In Cochrane, J.L. and Zeleny, M. (Eds.), Multiple Criteria Decision Making, University of South Carolina Press, Columbia, South Carolina, 1973, pp. 18-44.

39. Major, D.C., "Benefit-Cost Ratios for Projects in Multiple Objective Investment Programs", Water Resources Research, Vol. 5, No. 6, December 1969, pp. 1174-1178.

40. Marglin, S.A., Public Investment Criteria, MIT Press, Cambridge, Massachusetts, 1967.

41. Monarchi, D., Kisiel, C., and Duckstein, L., "Interactive Multi-Objective Programming in Water Resources: A Case Study", Water Resources Research, Vol. 9, No. 4, August 1973, pp. 837-850.

42. Montgomery, D.C. and Bettencourt, Jr., V.M., "Multiple Response Surface Methods in Computer Simulation", Working Paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1976.

43. Oppenheimer, K.R., "A Proxy Approach to Multi-Attribute Decision Making", Management Science, Vol. 24, No. 6, February 1978, pp. 675-689.

44. Phillips, D.T., Ravindran, A., and Solberg, J.J., Operations Research: Principles and Practice, John Wiley & Sons, New York, 1976.

45. Plane, D.R. and McMillan, C., Jr., Discrete Optimization, Prentice-Hall, New Jersey, 1971.

46. Polak, E., Computational Methods in Optimization, Academic Press, New York, 1971.

47. Roesner, L.A., Nichandros, H.M., and Shubinski, R.P., "A Model for Evaluating Runoff-Quality in Metropolitan Master Planning", ASCE Urban Water Resources Research Program, Technical Memorandum No. 23, April 1974.

48. Roy, B., "Problems and Methods with Multiple Objective Functions", Mathematical Programming, Vol. I, No. 2, November 1971, pp. 239-266.

49. Royden, H.L., Real Analysis, MacMillan Company, London, 1968.

50. Sen, A.K., Collective Choice and Social Welfare, Holden-Day, San Francisco, 1970.

51. Simmons, D.M., Nonlinear Programming for Operations Research, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

52. Soland, R.M., "Multicriteria Optimization: A General Characterization of Efficient Solutions", Rapport Technique EP 77-R-8, Ecole Polytechnique de Montreal, March 1977.

53. Ting, H.M., "Aggregation of Attributes for Multiattributed Utility Assessment", Technical Report No. 66, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 1971.

54. Tversky, A., "Elimination by Aspects: A Theory of Choice", Psychological Review, Vol. 79, 1972, pp. 281-299.

55. Tversky, A., "Choice by Elimination", Journal of Mathematical Psychology, Vol. 9, 1972, pp. 341-367.

56. U.S. Army Corps of Engineers, Hydrologic Engineering Center, "Urban Storm Water Runoff 'STORM'," Generalized Computer Program, 723-58-L2520, May 1974.

57. Wallenius, J., "Comparative Evaluation of Some Interactive Approaches to Multicriterion Optimization", Management Science, Vol. 21, No. 12, August 1975, pp. 1387-1396.

58. Wilde, D.J., "Optimization by the Method of Contour Tangents", American Institute of Chemical Engineers Journal, Vol. 9, No. 2, March 1963, pp. 186-190.

59. Wilde, D.J., Optimum Seeking Methods, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

60. Witzgall, C., "An All-Integer Programming Algorithm with Parabolic Constraints", Journal of the Society for Industrial and Applied Mathematics, Vol. 11, No. 4, December 1963, pp. 855-871.

61. Wolfe, P., "Convergence Theory in Nonlinear Programming", In Integer and Nonlinear Programming, Abadie, J. (Ed.), North-Holland Publishing Company, Amsterdam, 1970.

62. Zangwill, W.I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

63. Zionts, S., "Multiple Criteria Decision Making for Discrete Alternatives with Multiple Objectives", Working Paper No. 299, School of Management, State University of New York at Buffalo, February 1977.

64. Zionts, S. and Wallenius, J., "An Interactive Programming Method for Solving the Multiple Criteria Problem", Management Science, Vol. 22, No. 6, February 1976, pp. 652-663.

APPENDICES

APPENDIX A

AN AMALGAM OF DEFINITIONS AND LEMMAS

This appendix presents various lemmas, theorems and definitions. They are introduced here with the intention of standardizing notation and making it possible for a more lucid discussion throughout the main text. Lemmas peculiar to this thesis are proved in detail.

Definition (Convex Set)

A set $X \subset E^n$ is convex if and only if $\underline{x}^1, \underline{x}^2 \in X$ implies $\underline{y} = \lambda \underline{x}^1 + (1-\lambda)\underline{x}^2$ is also in X for any λ, $0 \leq \lambda \leq 1$.

Definition (Compact)

A compact set is one which is closed and bounded.

Definition (Concave)

Given a convex set X, a function U is concave on X if and only if $\underline{x}^1, \underline{x}^2 \in X$ implies $U(\lambda \underline{x}^1 + (1-\lambda)\underline{x}^2) \geq \lambda U(\underline{x}^1) + (1-\lambda) U(\underline{x}^2)$ where $0 \leq \lambda \leq 1$.

Definition (Increasing)

A function U is termed increasing if $y^1 < y^2$ implies $U(y^1) \leq U(y^2)$.

Lemma A.1

Let X_i, i=1,2,...,m, be convex sets. Then the set $X_m = \bigcap_{i=1}^{m} X_i$ is also convex.

Lemma A.2

Let f be a concave function. Then for any fixed scalar c, the set in E^n

$$F_c = \{ \underline{x} \mid f(\underline{x}) \geq c \}$$

is convex.

Lemma A.3

Let f_i, i=1,2,...,m, be concave functions on the convex set X. The set

$$F = \{ \underline{x} \mid f_i(\underline{x}) \geq c_i, \ \underline{x} \in X, \ i=1,2,...,m \}$$

is convex.

Lemma A.4

Let f_i, i=1,2,...,r, each be concave on a convex set X. If $\alpha_i \geq 0$, i=1,2,...,r, the function $g(\underline{x}) = \sum_{i=1}^{r} \alpha_i f_i(\underline{x})$ is concave on X.

Lemma A.5

If U is concave increasing on Y and each f_i is concave on X, then U is concave on X.

Lemma A.6 [62]

Let U be differentiable on E^n. Then U is concave on X if and only if $U(\underline{y}) \leq U(\underline{x}) + \nabla_{\underline{x}} U(\underline{x})^t \cdot (\underline{y} - \underline{x})$ for any $\underline{x}$ and $\underline{y}$ in X.

The next two lemmas explain some of the relationships between the decision space and the objective space. Preliminary to this, define

$$X_m \equiv \{ \underline{x} \mid \underline{x} \in \bigcap_{i=0}^{m} X^i \}$$

where $X^0 \equiv X$, a convex set,

$$X^1 \equiv \{ \underline{x} \mid \sum_{k=1}^{r} \alpha_k^1 f_k(\underline{x}) \geq c_1 \}$$

$$\vdots$$

$$X^m \equiv \{ \underline{x} \mid \sum_{k=1}^{r} \alpha_k^m f_k(\underline{x}) \geq c_m \}$$

where c_i, i=1,2,...,m, and $\alpha_k \geq 0$, k=1,2,...,r, are scalars; and f_k, k=1,2,...,r, are concave functions defined on the convex set X. Also define

$$F_m \equiv \{ \underline{f} \mid \underline{f} \in \bigcap_{i=0}^{m} F^i \}$$

where $F^0 \equiv Y = \{ \underline{f} \mid \underline{f} = \underline{f}(\underline{x}), \underline{x} \in X \}$

$$F^1 \equiv \{ \underline{f} \mid \sum_{k=1}^{r} \alpha_k^1 f_k \geq c_1 \}$$

$$\vdots$$

$$F^m \equiv \{ \underline{f} \mid \sum_{k=1}^{r} \alpha_k^m f_k \geq c_m \}.$$

Lemma A.7

$\underline{x} \in X_m \Rightarrow \underline{f}(\underline{x}) \in F_m$.

Proof.

$$\underline{x} \in X_m \Rightarrow \underline{x} \in \bigcap_{i=0}^{m} X^i$$

$$\Rightarrow \begin{cases} \underline{x} \in X \\ \sum_{k=1}^{r} \alpha_k^j f_k(\underline{x}) \geq c_j, \quad j=1,2,\ldots,m \end{cases}$$

$$\Rightarrow \begin{cases} f(\underline{x}) \in Y \\ f(\underline{x}) \in F^j, \quad j=1,2,\ldots,m \end{cases}$$

$$\Rightarrow \quad f(\underline{x}) \in \bigcap_{i=0}^{m} F^i$$

$$\Rightarrow \quad f(\underline{x}) \in F_m.$$

Q.E.D.

Lemma A.8

If $\underline{f}^* \in F_m$, then $\exists\, \underline{x} \in X$ such that $\underline{f}(\underline{x}) = \underline{f}^*$. Furthermore, for any $\underline{x} \in X$ for which $\underline{f}(\underline{x}) = \underline{f}^*$, $\underline{x} \in X_m$.

Proof.

Part I.

$$f^* \in F_m \Rightarrow f^* \in \bigcap_{i=0}^{m} F^i$$

$$\Rightarrow f^* \in F^o \equiv F$$

$$\Rightarrow \exists\, \underline{x} \in X \text{ such that } f(\underline{x}) = f^*.$$

Part II.

Let $\underline{x} \in X$ and $f(\underline{x}) = f^* \in F_m$. Then

$$\sum_{k=1}^{r} \alpha_k^j f_k(\underline{x}) \geq c_j, \quad j=1,2,\ldots,m$$

$$\Rightarrow \underline{x} \in \bigcap_{i=0}^{m} X^i \Rightarrow \underline{x} \in X_m.$$

Q.E.D.

Lemma A.9 [62]

Let $f:V \to \mathbb{R}$ be a continuous function, and $S \subset V$ where S is a compact set. Suppose

$$f(w) > 0, \quad \forall\, w \in S$$

then $\inf\{f(w) \mid w \in S\} > 0$.

Lemma A.10 [46]

Let $f: \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function, and $S \subset \mathbb{R}^n$ where S is a compact set. Then for each $\underline{z}' \in \mathbb{R}^n$ and for each $\delta > 0$, there exists an $\epsilon' > 0$ and a $\lambda_m > 0$ such

that for all $\underline{z} \in D(\underline{z}', \epsilon') = \{\underline{z} \mid \|\underline{z} - \underline{z}'\| \leq \epsilon'\}$ and for all $\underline{h} \in S$,

$$|\nabla_{\underline{z}} f(\underline{z} + \lambda \underline{h}) \cdot \underline{h} - \nabla_{\underline{z}} f(\underline{z}) \cdot \underline{h}| \leq \delta \quad \text{for all } \lambda \in [0, \lambda_m].$$

Theorem A.1 (Mean-Value Theorem)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function. Then, for any $\underline{z}, \underline{h} \in \mathbb{R}^n$ and for any $\lambda \in \mathbb{R}$,

$$f(\underline{z} + \lambda \underline{h}) = f(\underline{z}) + \lambda(\nabla_{\underline{\xi}} f(\underline{\xi}) \cdot \underline{h})$$

where $\xi \in [\underline{z}, \underline{z} + \lambda \underline{h}]$.

Condition. (Constraint Qualification) [51]

Let $\underline{x}^o$ be any feasible solution to

$$\text{Max} \quad U(\underline{x})$$
$$\text{subject to} \quad g_i(\underline{x}) \leq 0, \quad i=1,2,\ldots,m_k \ ;$$

and let $\underline{h}$ be any vector originating at $\underline{x}^o$ which satisfies

$$\nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} \leq 0 \quad \text{for all } i \in A(\underline{x}) \equiv \{i \mid g_i(\underline{x}) = 0, \ i=1,2,\ldots,m_k\}.$$

Then if the constraint qualification is to hold, there must exist an n-dimensional vector function $\underline{\varphi}(t)$ of a nonnegative real variable t such that

(a) $\underline{\varphi}(0) = \underline{x}^o$;

(b) $\underline{\varphi}(t) = \underline{x}$ is a feasible point for all $0 \le t \le T$, where T is some positive number; and

(c) $\lim_{t \to 0} \dfrac{\underline{\varphi}(t) - \underline{\varphi}(0)}{t} = \underline{h}$.

In essense, this condition ensures that every direction $\underline{h}$ satisfying $\nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} \le 0$ for all $i \in A(\underline{x})$ is a feasible direction.

Lemma A.11

Assume U to be concave increasing and differentiable on $C_k \equiv \{ \underline{x} \in \mathbb{R}^n \mid g_i(\underline{x}) \le 0, \quad i=1,2,\ldots,m_k \}$ where $g_i(\underline{x})$ are convex differentiable functions. Let $x \equiv \partial U / \partial f_1$ and $S \equiv \{ \underline{h} \in \mathbb{R}^n \mid |h_i| \le 1, \quad i=1,2,\ldots,n \}$. Then

$$\min_{\underline{h} \in S} \left(\max \{ -\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h},\ g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h}, \quad i=1,2,\ldots,m_k \} \right) \ge 0$$

$$\Leftrightarrow \min_{\underline{h} \in S} \max_{i \in A(\underline{x})} \{ -\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h},\ \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} \} \ge 0,$$

where $A(\underline{x}) \equiv \{ i \mid g_i(\underline{x}) = 0,\ i=1,2,\ldots,m_k \}$.

Proof.

Case 1. ($\Rightarrow$)

If $\min_{\underline{h} \in S} \max_{i \in A(\underline{x})} \{ -\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h},\ \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} \} < 0,$

then $-\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} < 0$ and

$$\nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} < 0 \quad \text{for } i \in A(\underline{x}).$$

Therefore, $g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} < 0$ for $i \in A(\underline{x})$.

But, for $i \notin A(\underline{x})$, $g_i(\underline{x}) < 0$. Accordingly, $\exists\, \underline{h} \in S$ such that $g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} < 0$.

Thus,

$$\min_{\underline{h} \in S} \left(\max \{-\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h},\ g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h},\quad i=1,2,\ldots,m_k \}\right) < 0.$$

The first case is established.

Case 2. ($\Leftarrow$)

If $$\min_{\underline{h} \in S} \left(\max \{-\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h},\ g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h},\ i=1,2,\ldots,m_k \}\right) < 0,$$

then $-\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h} < 0$ and $g_i(\underline{x}) + \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} < 0$, for some $\underline{h} \in S$. For $i \in A(\underline{x})$, $\nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h} < 0$.

Thus,

$$\min_{\underline{h} \in S} \max_{i \in A(\underline{x})} \{-\alpha \nabla_{\underline{x}} U(\underline{x})^t \cdot \underline{h},\ \nabla_{\underline{x}} g_i(\underline{x})^t \cdot \underline{h}\} < 0.$$

This establishes the second case and the lemma.

APPENDIX B

METHOD OF CENTERS

In the decision space a compact convex set encloses what is being termed the best-compromise solution. The goal is to divide this space by means of a cut. Clearly, a cut near the "center" of the eligible region would be advantageous since a significant portion of the region would be eliminated each time.

Wilde [58] proposes this type of an idea in his article "Optimization by the Method of Contour Tangents". In order to find the middle of the feasible region, he suggests locating one of the following four points: the midpoint, the minimax point, the center of volume, or the centroid.

Because these points are computationally too unreasonable to locate with respect to the problem being studied, another means must be used. A technique following the lines of those known as "methods of centers" is remarkably applicable in the context of the problem under study. This appendix is devoted to describing Huard's Modified Method of Centers [29, 30] as presented in Polak [46].

B.1. Preliminary Observations

In the modified method of centers, the problem is to

$$\begin{aligned} &\text{minimize} \quad f_0(\underline{x}) \\ &\text{subject to} \quad f_i(\underline{x}) \leq 0, \quad i=1,2,\ldots,m \end{aligned} \tag{B.1}$$

where f_i: $R^m \to R$, $i=0,1,2,\ldots,m$, are continuously differentiable functions.

Two assumptions are made on the set

$$C = \{ \underline{z} \mid f_i(\underline{z}) \leq 0, \quad i=0,1,2,\ldots,m \} \tag{B.2}$$

and on the objective function $f_0(\underline{z})$.

A1. There is a point $\underline{z}^0 \in C$ such that the set

$$C'(\underline{z}^0) = \{ \underline{z} \mid f_0(\underline{z}) - f_0(\underline{z}^0) \leq 0, \quad f_i(\underline{z}) \leq 0, \quad i=1,2,\ldots,m \} \tag{B.3}$$

is compact and has an interior.

A2. a) The set C in (B.2) has an interior and the closure of the interior of C is equal to C.

b) For every $\underline{z}$ in the interior of C, $f_i(\underline{z}) < 0$, $i=1,2,\ldots,m$.

Since the object is to find the middle of the feasible region at each step, a distance measure is necessary.

Definition (Distance)

Given any point $\underline{x}$ in a convex set C, the distance d of $\underline{x}$ to the boundary of C is a function which is continuous and possesses the following properties:

1) $d = 0$ for all points $\underline{x}$ on the boundary of C,

2) $d > 0$ for all points $\underline{x}$ in the interior of C.

Definition (Center)

Given a convex set C and a distance function d defined on this set, a point $\underline{x}$ in C which maximizes d over the set C is called the center of the set.

Given that the current objective value is $f_0(\underline{z}')$, let d: $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be defined by

$$d(\underline{z},\underline{z}') = \max \{ f_0(\underline{z}) - f_0(\underline{z}'),\ f_i(\underline{z}),\quad i=1,2,\ldots,m \}. \tag{B.4}$$

The function d is a non-positive function for which -d satisfies the requirements of a distance function. Although the choice of a distance function is not unique [8, 29], the one defined above is fitting for the type of situation which is encountered in this report. Note, also, that if the functions f_i, i=0,1,2,...,m, are all convex, then the distance function d is also convex for any $\underline{z} \in C$.

Lemma B.1

If $\hat{\underline{z}}$ is optimal to (B.1), then

$$\min_{\underline{h} \in S} \left(\max \{ \nabla_{\underline{z}} f_0(\hat{\underline{z}})^t \cdot \underline{h};\ \nabla_{\underline{z}} f_i(\hat{\underline{z}})^t \cdot \underline{h},\quad i=1,2,\ldots,m\}\right) = 0$$

$$\text{where } S = \{ \underline{h} \mid |h_i| \le 1,\quad i=1,2,\ldots,n,\ \underline{h} \in \mathbb{R}^n \}.$$

The vector $\underline{h}$ is interpreted to be a direction vector. By minimizing the left-hand-side expression over all $\underline{h} \in S$, the resultant $\underline{h}$ is also a usable direction.

A more practical version of Lemma B.1 can be stated by defining $M(\underline{z})$, for any $\underline{h} \in S$, as

$$M(\underline{z}) = \max \{ \nabla_{\underline{z}} f_0(\underline{z})^t \cdot \underline{h} ; \; f_i(\underline{z}) + \nabla_{\underline{z}} f_i(\underline{z})^t \cdot \underline{h} ,$$

$$i=1,2,\ldots,m\} \qquad (B.5)$$

Lemma B.2

If $\hat{\underline{z}}$ is optimal to (B.1), then $M(\hat{\underline{z}}) = 0$ where $(M(\underline{z}), \underline{h}) \in R^{n+1}$ is characterized by the linear programming problem:

minimize M

$$\text{subject to:} \quad -M + \nabla_{\underline{z}} f_0(\underline{z})^t \cdot \underline{h} \leq 0 \qquad (B.6)$$

$$-M + f_i(\underline{z}) + \nabla_{\underline{z}} f_i(\underline{z})^t \cdot \underline{h} \leq 0, \quad i=1,2,\ldots,m$$

$$-1 \leq h_j \leq 1 \quad , \quad j=1,2,\ldots,n$$

Three points should be emphasized. First, since the functions f_i, $i=0,1,2,\ldots,m$, are continuously differentiable, M is the maximum of a set of continuous functions. Therefore, it must be a continuous function. Secondly, for any point $\underline{z} \in C$, $M(\underline{z}) \leq 0$. Finally, the vector $\underline{h}$ might not be uniquely defined by the LP problem in (B.6). Nevertheless, any optimal vector $\underline{h}$ is acceptable.

B.2. Huard's Modified Method of Centers [29]

Huard proposes the following modified method of centers which, according to Polak, is the most efficient version.

Algorithm

Step 0. Determine a point $\underline{z}^0 \in C$, and set $i=0$.

Step 1. Set $\underline{z} = \underline{z}^i$.

Step 2. Solve (B.6) to obtain $(M(\underline{z}),\underline{h})$.

Step 3. If $M(\underline{z}) = 0$, stop; otherwise, continue.

Step 4. Determine $\mu(\underline{z})$, the smallest positive scalar such that

$$d(\underline{z} + \mu(\underline{z})\underline{h}, \underline{z}) = \min \{ d(\underline{z} + \mu \underline{h}, \underline{z}) \mid \mu \geq 0 \}$$

where d is defined as in (B.4).

Step 5. Set $\underline{z}^{i+1} = \underline{z} + \mu(\underline{z})\underline{h}$, i=i+1, and return to Step 1.

Figure B.1 illustrates what occurs on one iteration of Huard's algorithm. Starting from a point $\underline{z}^i$ in C, the usable direction $\underline{h}$ is determined by the LP problem (B.6). The successor point, $\underline{z}^{i+1}$, is then ascertained by minimizing along $\underline{h}$ the maximum distance from all the boundaries of C, and the process continues. It is this type of reasoning which is used in the Tradeoff Cutting Plane algorithm to locate the next point in decision space.

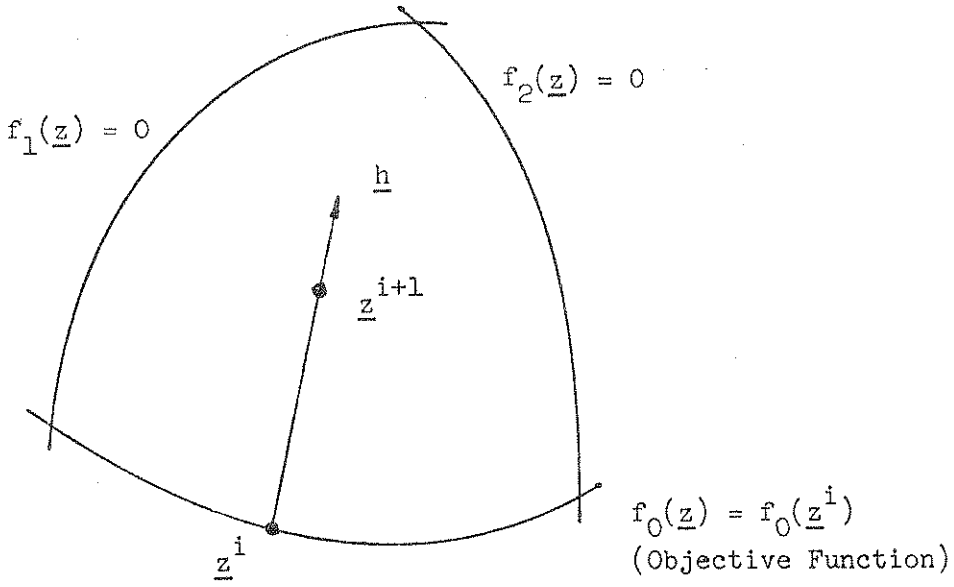


Figure B.1. Locating $\underline{z}^{i+1}$ from $\underline{z}^i$

APPENDIX C

PROGRAM LISTING

```
      PROGRAM TCP (INPUT,TERM,OUTPUT,TAPE5=INPUT,TAPE8=TERM,TAPE6=OUTPUTTCP    10
     1)                                                                 TCP    20
      COMMON /TCOM1/ NX,NCF,NC,NACC                                     TCP    30
      COMMON /TCOM2/ TRADE(20,10),HOLD(20),H(20)                        TCP    40
      COMMON /TCOM3/ CCL(41,121),C(121),CBAR(121)                       TCP    50
      DIMENSION X(20), XI(20), XP(20), XOLD(20), Z(20), W(10)           TCP    60
      DIMENSION F(10), OLDF(20,10), NAME(10), BB(10), GF(200), G(20), GGTCP    70
     1(39)                                                              TCP    80
      DIMENSION GIN(20), GMIN(20), GMAX(20), TCMAX(20)                  TCP    90
      DATA Z/20*0./,GMIN/20*9.E30/,GMAX/20*0./,TCMAX/20*0./             TCP   100
C                                                                       TCP   110
C                                                                       TCP   120
C     THIS SECTION INPUTS INITIAL DATA TO START THE TRADEOFF            TCP   130
C     CUTTING PLANE ALGORITHM.                                          TCP   140
C                                                                       TCP   150
C                                                                       TCP   160
      WRITE (6,105)                                                     TCP   170
  105 FORMAT (1H1)                                                      TCP   180
      WRITE (6,110)                                                     TCP   190
      WRITE (8,110)                                                     TCP   200
  110 FORMAT (//20X, 32HTRADEOFF CUTTING PLANE ALGORITHM,/,13X, 47HFOR ITCP   210
     1NTERACTIVE MULTIPLE OBJECTIVE OPTIMIZATION,//)                    TCP   220
      WRITE (6,115)                                                     TCP   230
      WRITE (8,115)                                                     TCP   240
  115 FORMAT (5X, 39HHOW MANY OBJECTIVE FUNCTIONS ARE THERE?)           TCP   250
      CALL RI (NOF)                                                     TCP   260
      WRITE (6,120) NOF                                                 TCP   270
  120 FORMAT (1X,I2)                                                    TCP   280
      WRITE (6,125)                                                     TCP   290
      WRITE (8,125)                                                     TCP   300
  125 FORMAT (5X, 45HIN TEN LETTERS OR LESS, INPUT EACH OBJECTIVE , 14HFTCP   310
     1UNCTION NAME.)                                                    TCP   320
      DO 140 I=1,NOF                                                    TCP   330
         WRITE (6,130) I                                                TCP   340
         WRITE (8,130) I                                                TCP   350
  130 FORMAT (5X, 35HWHAT IS THE NAME ASSOCIATED WITH F(,I1,  2H)?)      TCP   360
         CALL RC (NAME(I))                                              TCP   370
         WRITE (6,135) NAME(I)                                          TCP   380
  135 FORMAT (1X,A10)                                                   TCP   390
  140 CONTINUE                                                          TCP   400
      WRITE (6,145)                                                     TCP   410
      WRITE (8,145)                                                     TCP   420
  145 FORMAT (5X, 31HHOW MANY CONSTRAINTS ARE THERE?)                   TCP   430
      CALL RI (NC)                                                      TCP   440
      WRITE (6,150) NC                                                  TCP   450
  150 FORMAT (1X,I2)                                                    TCP   460
      WRITE (6,155)                                                     TCP   470
      WRITE (8,155)                                                     TCP   480
  155 FORMAT (5X, 41HHOW MANY INDEPENDENT VARIABLES ARE THERE?)         TCP   490
      CALL RI (NX)                                                      TCP   500
      WRITE (6,160) NX                                                  TCP   510
  160 FORMAT (1X,I2)                                                    TCP   520
  165 WRITE (6,170)                                                     TCP   530
      WRITE (8,170)                                                     TCP   540
  170 FORMAT (5X, 49HINPUT INITIAL VALUE OF THE INDEPENDENT VARIABLES.) TCP   550
      DO 185 I=1,NX                                                     TCP   560
         WRITE (6,175) I                                                TCP   570
         WRITE (8,175) I                                                TCP   580
  175 FORMAT (1X,  4H  X(,I2,  4H) = )                                  TCP   590
         CALL RR (X(I))                                                 TCP   600
```

```
      WRITE (6,180) X(I)                                                TCP  610
  180 FORMAT (1X,F10.4)                                                 TCP  620
      XI(I)=X(I)                                                        TCP  630
  185 CONTINUE                                                          TCP  640
      CALL GFXNS (X,G)                                                  TCP  650
      DO 205 K=1,NC                                                     TCP  660
      GIN(K)=-G(K)                                                      TCP  670
      IF (G(K).LE.0.) GO TO 205                                         TCP  680
      WRITE (6,190)                                                     TCP  690
      WRITE (8,190)                                                     TCP  700
  190 FORMAT (/5X, 30HTHESE VALUES ARE NOT FEASIBLE.,/,5X, 43HDO YOU WISTCP  710
     1H TO TRY AGAINS  IF SO, TYPE Y.  , 18HOTHERWISE, TYPE N.)         TCP  720
      CALL RC (ITRY)                                                    TCP  730
      WRITE (6,195) ITRY                                                TCP  740
  195 FORMAT (1X,A1)                                                    TCP  750
      IF (ITRY.EQ.1HY) GO TO 165                                        TCP  760
      WRITE (6,200)                                                     TCP  770
      WRITE (8,200)                                                     TCP  780
  200 FORMAT (5X, 33HTERMINATION DUE TO INFEASIBILITY.)                 TCP  790
      GO TO 955                                                         TCP  800
  205 CONTINUE                                                          TCP  810
      WRITE (6,210)                                                     TCP  820
      WRITE (8,210)                                                     TCP  830
  210 FORMAT (/5X, 39HYOU ARE LOCATED IN THE FEASIBLE REGION.,/)        TCP  840
      WRITE (6,215)                                                     TCP  850
      WRITE (8,215)                                                     TCP  860
  215 FORMAT (5X, 50HINPUT THE STEP SIZE FOR THE GOLDEN SECTION SEARCH.)TCP  870
      CALL RR (RHO)                                                     TCP  880
      WRITE (6,220) RHO                                                 TCP  890
  220 FORMAT (1X,F10.8)                                                 TCP  900
      WRITE (6,225)                                                     TCP  910
      WRITE (8,225)                                                     TCP  920
  225 FORMAT (5X, 49HINPUT THE INITIAL ERROR TOLERANCE FOR THE GOLDEN , TCP  930
     1 15HSECTION SEARCH.)                                              TCP  940
      CALL RR (ERR)                                                     TCP  950
      WRITE (6,230) ERR                                                 TCP  960
  230 FORMAT (1X,F10.8)                                                 TCP  970
      WRITE (6,235)                                                     TCP  980
      WRITE (8,235)                                                     TCP  990
  235 FORMAT (5X, 46HDO YOU WISH TO USE THE GRADIENT APPROXIMATION ,  9HTCP 1000
     1ROUTINES,/5X, 34HIF SO, TYPE Y.  OTHERWISE, TYPE N.)              TCP 1010
      CALL RC (IGRAD)                                                   TCP 1020
      WRITE (6,240) IGRAD                                               TCP 1030
  240 FORMAT (1X,A1)                                                    TCP 1040
      IF (IGRAD.EQ.1HA) GO TO 255                                       TCP 1050
      WRITE (6,245)                                                     TCP 1060
      WRITE (8,245)                                                     TCP 1070
  245 FORMAT (5X, 48HINPUT THE CENTRAL DIFFERENCING PARAMETER FOR CAL, 1TCP 1080
     19HCULATING GRADIENTS.)                                            TCP 1090
      CALL RR (DIF)                                                     TCP 1100
      WRITE (6,250) DIF                                                 TCP 1110
  250 FORMAT (1X,F10.8)                                                 TCP 1120
  255 WRITE (6,260)                                                     TCP 1130
      WRITE (8,260)                                                     TCP 1140
  260 FORMAT (5X, 39HDO YOU WISH TO USE THE TRADEOFF SCHEMES,/5X, 34HIF TCP 1150
     1SO, TYPE Y.  OTHERWISE, TYPE N.)                                  TCP 1160
      CALL RC (ITRAD)                                                   TCP 1170
      WRITE (6,265) ITRAD                                               TCP 1180
  265 FORMAT (1X,A1)                                                    TCP 1190
C                                                                       TCP 1200
C                                                                       TCP 1210
```

```
C     THIS SECTION PRESENTS THE DECISION MAKER WITH ALTERNATIVES AND     TCP 1220
C     READS HIS TRADEOFF INPUTS.                                         TCP 1230
C                                                                        TCP 1240
C                                                                        TCP 1250
      NACC=0                                                             TCP 1260
      W(1)=1.                                                            TCP 1270
      IC=1                                                               TCP 1280
      L=1                                                                TCP 1290
      T=1.                                                               TCP 1300
      CALL FFXNS (X,F)                                                   TCP 1310
      DO 270 I=1,NX                                                      TCP 1320
  270 XOLD(I)=X(I)                                                       TCP 1330
      WRITE (6,275)                                                      TCP 1340
      WRITE (8,275)                                                      TCP 1350
  275 FORMAT (5X, 42HTHE INITIAL OBJECTIVE FUNCTION VALUES ARE:)         TCP 1360
      DO 285 I=1,NOF                                                     TCP 1370
         WRITE (6,280) NAME(I),I,F(I)                                    TCP 1380
         WRITE (8,280) NAME(I),I,F(I)                                    TCP 1390
  280 FORMAT (1X,A10,10X,  2HF(,I1,  1H),5X,F20.5)                       TCP 1400
         OLDF(1,I)=F(I)                                                  TCP 1410
  285 CONTINUE                                                           TCP 1420
  290 WRITE (6,295)                                                      TCP 1430
      WRITE (8,295)                                                      TCP 1440
  295 FORMAT (/5X, 27HINPUT PERTURBATION OF F(1).)                       TCP 1450
      CALL RR (PFONE)                                                    TCP 1460
      WRITE (6,300) PFONE                                                TCP 1470
  300 FORMAT (1X,F15.7)                                                  TCP 1480
      BB(1)=F(1)+PFONE                                                   TCP 1490
      DO 305 JB=2,NOF                                                    TCP 1500
         BB(JB)=F(JB)                                                    TCP 1510
  305 CONTINUE                                                           TCP 1520
      DO 375 KT=2,NOF                                                    TCP 1530
         WRITE (6,310) KT                                                TCP 1540
         WRITE (8,310) KT                                                TCP 1550
  310 FORMAT (5X, 24HINPUT PERTURBATION OF F(,I1,  2H).)                 TCP 1560
         CALL RR (PERT)                                                  TCP 1570
         WRITE (6,315) PERT                                              TCP 1580
  315 FORMAT (1X,F15.7)                                                  TCP 1590
         DELTA=(PFONE*PFONE)/PERT                                        TCP 1600
         IF (KT.EQ.2) GO TO 320                                          TCP 1610
         BB(KT-1)=F(KT-1)                                                TCP 1620
  320    BB(KT)=F(KT)                                                    TCP 1630
  325    BB(KT)=BB(KT)-DELTA                                             TCP 1640
         IF (ITRAD.EQ.1HN) GO TO 370                                     TCP 1650
         WRITE (6,330)                                                   TCP 1660
         WRITE (8,330)                                                   TCP 1670
  330 FORMAT (30X,1HA,24X,1HB)                                           TCP 1680
      DO 340 NW=1,NOF                                                    TCP 1690
         WRITE (6,335) NAME(NW),F(NW),BB(NW)                             TCP 1700
         WRITE (8,335) NAME(NW),F(NW),BB(NW)                             TCP 1710
  335 FORMAT (1X,A10,5X,F20.5,5X,F20.5)                                  TCP 1720
  340    CONTINUE                                                        TCP 1730
C                                                                        TCP 1740
C                                                                        TCP 1750
C     THIS SECTION ADJUSTS THE ALTERNATIVES PRESENTED TO THE DECISION    TCP 1760
C     MAKER UNTIL HE IS INDIFFERENT.                                     TCP 1770
C                                                                        TCP 1780
C                                                                        TCP 1790
        WRITE (6,345)                                                    TCP 1800
        WRITE (8,345)                                                    TCP 1810
  345 FORMAT (5X, 53HWHICH DO YOU PREFER?  IF YOU ARE INDIFFERENT, TYPE  TCP 1820
```

```
     1I.)                                                              TCP 1830
          CALL RC (IPREF)                                              TCP 1840
          WRITE (6,350) IPREF                                          TCP 1850
  350 FORMAT (1X,A1)                                                   TCP 1860
          IF (IPREF.EQ.1HI) GO TO 370                                  TCP 1870
          IF (IPREF.EQ.1HB) GO TO 325                                  TCP 1880
C                                                                      TCP 1890
C                                                                      TCP 1900
C     PREFER A                                                         TCP 1910
C                                                                      TCP 1920
C                                                                      TCP 1930
  355     DELTA=DELTA/2.                                               TCP 1940
          BB(KT)=BB(KT)+DELTA                                          TCP 1950
C                                                                      TCP 1960
C                                                                      TCP 1970
C     INPUT PERFERENCE                                                 TCP 1980
C                                                                      TCP 1990
C                                                                      TCP 2000
  360     WRITE (6,330)                                                TCP 2010
          WRITE (8,330)                                                TCP 2020
          DO 365 JW=1,NCF                                              TCP 2030
             WRITE (6,335) NAME(JW),F(JW),BB(JW)                       TCP 2040
             WRITE (8,335) NAME(JW),F(JW),BB(JW)                       TCP 2050
  365     CONTINUE                                                     TCP 2060
          WRITE (6,345)                                                TCP 2070
          WRITE (8,345)                                                TCP 2080
          CALL RC (IPREF)                                              TCP 2090
          WRITE (6,350) IPREF                                          TCP 2100
          IF (IPREF.EQ.1HI) GO TO 370                                  TCP 2110
          IF (IPREF.EQ.1HA) GO TO 355                                  TCP 2120
C                                                                      TCP 2130
C                                                                      TCP 2140
C     PREFER B                                                         TCP 2150
C                                                                      TCP 2160
C                                                                      TCP 2170
          DELTA=DELTA/2.                                               TCP 2180
          BB(KT)=BB(KT)-DELTA                                          TCP 2190
          GO TO 360                                                    TCP 2200
C                                                                      TCP 2210
C                                                                      TCP 2220
C     THIS SECTION COMPUTES THE TRADEOFF VALUES.                       TCP 2230
C                                                                      TCP 2240
C                                                                      TCP 2250
  370     W(KT)=(F(KT)-BB(KT))/PFONE                                   TCP 2260
  375 CONTINUE                                                         TCP 2270
      NACC=NACC+1                                                      TCP 2280
      IF (ITRAD.EQ.1HN) GO TO 385                                      TCP 2290
      WRITE (6,380)                                                    TCP 2300
      WRITE (8,380)                                                    TCP 2310
  380 FORMAT (5X, 18HTHE TRADEOFFS ARE:)                               TCP 2320
  385 DO 400 LT=1,NCF                                                  TCP 2330
          IF (ITRAC.EQ.1HN) GO TO 395                                  TCP 2340
          WRITE (6,390) NAME(LT),W(LT)                                 TCP 2350
          WRITE (8,390) NAME(LT),W(LT)                                 TCP 2360
  390 FORMAT (1X,A10,5X,F10.5)                                         TCP 2370
  395     TRADE(NACC,LT)=W(LT)                                         TCP 2380
  400 CONTINUE                                                         TCP 2390
C                                                                      TCP 2400
C                                                                      TCP 2410
C     THIS SECTION CHECKS TO SEE IF THE DECISION MAKER IS IN THE       TCP 2420
C     PROMISING REGION.  IF REQUESTED, IT ALSO APPROXIMATES THE        TCP 2430
```

```
C     GRADIENT OF EACH OBJECTIVE FUNCTION USING A CENTRAL DIFFERENCING  TCP 2440
C     SCHEME.                                                           TCP 2450
C                                                                       TCP 2460
C                                                                       TCP 2470
      CALL HOLDC (NACC,F)                                               TCP 2480
      IF (IGRAD.EQ.1HY) GO TO 405                                       TCP 2490
      CALL FGRAD (X,GF)                                                 TCP 2500
      GO TO 430                                                         TCP 2510
  405 DO 410 I=1,NX                                                     TCP 2520
  410 XP(I)=X(I)                                                        TCP 2530
      DO 425 J=1,NX                                                     TCP 2540
         XP(J)=X(J)+DIF                                                 TCP 2550
         CALL FFXNS (XF,F)                                              TCP 2560
         DO 415 I=1,NCF                                                 TCP 2570
            IK=NX*I-NX+J                                                TCP 2580
  415    GF(IK)=F(I)                                                    TCP 2590
         XP(J)=X(J)-DIF                                                 TCP 2600
         CALL FFXNS (XF,F)                                              TCP 2610
         DO 420 I=1,NCF                                                 TCP 2620
            IK=NX*I-NX+J                                                TCP 2630
  420    GF(IK)=(GF(IK)-F(I))/(2.*DIF)                                  TCP 2640
         XP(J)=X(J)                                                     TCP 2650
  425 CONTINUE                                                          TCP 2660
      CALL FFXNS (X,F)                                                  TCP 2670
  430 IF (IC.EQ.1) GO TO 510                                            TCP 2680
C                                                                       TCP 2690
C                                                                       TCP 2700
C     TEST IF THERE IS GUARANTEED IMPROVEMENT.                          TCP 2710
C                                                                       TCP 2720
C                                                                       TCP 2730
      R=0.                                                              TCP 2740
      DO 440 K=1,NX                                                     TCP 2750
         ROC=0.                                                         TCP 2760
         DO 435 I=1,NCF                                                 TCP 2770
            INX=NX*I-NX+K                                               TCP 2780
  435    ROC=ROC+W(I)*GF(INX)                                           TCP 2790
  440 R=R+ROC*(-T)*Z(K)                                                 TCP 2800
      IF (R.LT.0.) GO TO 490                                            TCP 2810
C                                                                       TCP 2820
C                                                                       TCP 2830
C     TEST IF FORMER DECISION POINT SATISFIES THIS NEW CUT.             TCP 2840
C     IF NOT, THIS NEW POINT IS A PREFERRED POINT.                      TCP 2850
C                                                                       TCP 2860
C                                                                       TCP 2870
      CALL FFXNS (XOLD,F)                                               TCP 2880
      GOX=0.0                                                           TCP 2890
      DO 445 IJ=1,NOF                                                   TCP 2900
         GOX=GOX-W(IJ)*F(IJ)                                            TCP 2910
  445 CONTINUE                                                          TCP 2920
      GOX=GOX-HOLD(NACC)                                                TCP 2930
      IF (GOX.GT.0.0) GO TO 485                                         TCP 2940
C                                                                       TCP 2950
C                                                                       TCP 2960
C     TOO LARGE A STEP SIZE MAY HAVE BEEN TAKEN.                        TCP 2970
C     THE CUT IS RETAINED, BUT THE STEP SIZE IS REDUCED.                TCP 2980
C                                                                       TCP 2990
C                                                                       TCP 3000
      WRITE (6,450)                                                     TCP 3010
      WRITE (8,450)                                                     TCP 3020
  450 FORMAT (/5X, 36HPOSSIBLY TOO LARGE A STEP WAS TAKEN.,/5X, 39HONE-HTCP 3030
     1ALF THE DISTANCE IS NOW ATTEMPTED.)                               TCP 3040
```

```
      T=T/2.                                                              TCP 3050
      WRITE (6,455)                                                       TCP 3060
      WRITE (8,455)                                                       TCP 3070
  455 FORMAT (/5X, 36HTHE NEW ONE-HALF STEP SIZE POINT IS:)               TCP 3080
      DO 465 K=1,NX                                                       TCP 3090
         X(K)=XOLD(K)+(T*Z(K))                                            TCP 3100
         WRITE (6,460) K,X(K)                                             TCP 3110
         WRITE (8,460) K,X(K)                                             TCP 3120
  460 FORMAT (15X,  2HX(,I2,  4H) = ,F10.4)                               TCP 3130
  465 CONTINUE                                                            TCP 3140
      CALL FFXNS (X,F)                                                    TCP 3150
      WRITE (6,470)                                                       TCP 3160
      WRITE (8,470)                                                       TCP 3170
  470 FORMAT (/5X, 43HTHE OBJECTIVE FUNCTION VALUES FOR THIS ONE-, 25HHATCP 3180
     1LF STEP SIZE POINT ARE:)                                            TCP 3190
      DO 480 I=1,NOF                                                      TCP 3200
         WRITE (6,475) NAME(I),I,F(I)                                     TCP 3210
         WRITE (8,475) NAME(I),I,F(I)                                     TCP 3220
  475 FORMAT (1X,A10,10X,  2HF(,I1,  1H),5X,F20.5)                        TCP 3230
         OLDF(NACC+1,I)=F(I)                                              TCP 3240
  480 CONTINUE                                                            TCP 3250
      GO TO 290                                                           TCP 3260
C                                                                         TCP 3270
C                                                                         TCP 3280
C     THE DECISION MAKER IS AT A PREFERRED POINT.                         TCP 3290
C                                                                         TCP 3300
C                                                                         TCP 3310
  485 CALL FFXNS (X,F)                                                    TCP 3320
  490 WRITE (6,495)                                                       TCP 3330
      WRITE (8,495)                                                       TCP 3340
  495 FORMAT (/5X, 36HYOUR NEW POINT IS A PREFERRED POINT.)               TCP 3350
C                                                                         TCP 3360
C                                                                         TCP 3370
C     ASK WHETHER TO TERMINATE THE PROCESS OR NOT.                        TCP 3380
C                                                                         TCP 3390
C                                                                         TCP 3400
      WRITE (6,500)                                                       TCP 3410
      WRITE (8,500)                                                       TCP 3420
  500 FORMAT (/5X, 33HIF YOU WISH TO TERMINATE, TYPE T., 20H  OTHERWISE,TCP 3430
     1 TYPE C.)                                                           TCP 3440
      CALL RC (ITC)                                                       TCP 3450
      WRITE (6,505) ITC                                                   TCP 3460
  505 FORMAT (1X,A1)                                                      TCP 3470
      IF (ITC.EQ.1HT) GO TO 720                                           TCP 3480
C                                                                         TCP 3490
C                                                                         TCP 3500
C     THIS SECTION INPUTS THE COEFFICIENTS OF THE SUBOPTIMIZATION         TCP 3510
C     LINEAR PROGRAMMING PROBLEM WHICH REMAIN FIXED ON EACH ITERATION     TCP 3520
C     OF THE TRADEOFF CUTTING PLANE ALGORITHM.                            TCP 3530
C                                                                         TCP 3540
C                                                                         TCP 3550
  510 DO 525 IR=1,NX                                                      TCP 3560
         DO 520 ICOL=1,NX                                                 TCP 3570
            IF (IR.EQ.ICOL) GO TO 515                                     TCP 3580
            COL(IR,ICOL)=0.                                               TCP 3590
            NK=NX+ICOL                                                    TCP 3600
            COL(IR,NK)=0.                                                 TCP 3610
            NI=NX+IR                                                      TCP 3620
            COL(NI,ICOL)=0.                                               TCP 3630
            COL(NI,NK)=0.                                                 TCP 3640
            GO TO 520                                                     TCP 3650
```

```
  515         COL(IR,ICCL)=-1.                                          TCP 3660
              NK=NX+ICOL                                                TCP 3670
              COL(IR,NK)=1.0                                            TCP 3680
              NI=NX+IR                                                  TCP 3690
              COL(NI,ICCL)=1.                                           TCP 3700
              COL(NI,NK)=-1.0                                           TCP 3710
  520      CONTINUE                                                     TCP 3720
  525   CONTINUE                                                        TCP 3730
        NTX=2*NX                                                        TCP 3740
        DO 530 I=1,NTX                                                  TCP 3750
  530   COL(2*NX+1,I)=0.                                                TCP 3760
        DO 535 I=1,NTX                                                  TCP 3770
           C(I)=-1.                                                     TCP 3780
  535   CBAR(I)=-1.                                                     TCP 3790
C                                                                       TCP 3800
C                                                                       TCP 3810
C       THIS SECTION DETERMINES THE COEFFICIENTS OF THE SUBOPTIMIZATION  TCP 3820
C       LINEAR PROGRAMMING PROBLEM WHICH CHANGE ON EACH ITERATION OF THE TCP 3830
C       TRADEOFF CUTTING PLANE ALGORITHM.  INCLUDED IN THIS SECTION IS   TCP 3840
C       ALSO A CENTRAL DIFFERENCING SCHEME TO APPROXIMATE, IF SO         TCP 3850
C       REQUESTED, THE GRADIENT OF EACH CONSTRAINT.  FURTHERMORE, THIS   TCP 3860
C       SECTION CHECKS IF THE ORIGINAL CONSTRAINTS EXTEND THEIR PREVIOUS TCP 3870
C       MIN OR MAX VALUES.                                               TCP 3880
C                                                                       TCP 3890
C                                                                       TCP 3900
        IF (IGRAD.EQ.1HY) GO TO 550                                     TCP 3910
        CALL GGRAD (X,GG)                                               TCP 3920
        DO 545 J=1,NX                                                   TCP 3930
           JJ=NX+J                                                      TCP 3940
           DO 540 I=1,NC                                                TCP 3950
              IR=2*NX+I                                                 TCP 3960
              IG=NX*(IR-2*NX)-NX+J                                      TCP 3970
              COL(J,IR)=-GG(IG)                                         TCP 3980
              COL(JJ,IR)=GG(IG)                                         TCP 3990
  540      CONTINUE                                                     TCP 4000
  545   CONTINUE                                                        TCP 4010
        GO TO 575                                                       TCP 4020
  550   DO 555 I=1,NX                                                   TCP 4030
  555   XP(I)=X(I)                                                      TCP 4040
        DO 570 J=1,NX                                                   TCP 4050
           JJ=NX+J                                                      TCP 4060
           XP(J)=X(J)+DIF                                               TCP 4070
           CALL GFXNS (XP,G)                                            TCP 4080
           DO 560 I=1,NC                                                TCP 4090
              IR=2*NX+I                                                 TCP 4100
  560      COL(J,IR)=G(I)                                               TCP 4110
           XP(J)=X(J)-DIF                                               TCP 4120
           CALL GFXNS (XP,G)                                            TCP 4130
           DO 565 I=1,NC                                                TCP 4140
              IR=2*NX+I                                                 TCP 4150
              CCL(J,IR)=-(COL(J,IR)-G(I))/(2.*DIF)                      TCP 4160
  565      COL(JJ,IR)=-CCL(J,IR)                                        TCP 4170
           XP(J)=X(J)                                                   TCP 4180
  570   CONTINUE                                                        TCP 4190
  575   CALL GFXNS (X,G)                                                TCP 4200
        DO 595 I=1,NC                                                   TCP 4210
           IK=2*NX+I                                                    TCP 4220
           IF (-G(I).LE.GMAX(I)) GO TO 580                              TCP 4230
           GMAX(I)=-G(I)                                                TCP 4240
  580      IF (-G(I).GE.GMIN(I)) GO TO 585                              TCP 4250
           GMIN(I)=-G(I)                                                TCP 4260
```

```
  585     COL(2*NX+1,IK)=-1.0                                              TCP 4270
          C(IK)=G(I)                                                       TCP 4280
          CBAR(IK)=G(I)                                                    TCP 4290
          IF (C(IK).LE.0.0) GO TO 595                                      TCP 4300
          WRITE (6,590) IK                                                 TCP 4310
          WRITE (8,590) IK                                                 TCP 4320
  590 FORMAT (//1X,I3, 43H ELEMENT OF THE LP COST VECTOR IS POSITIVE.,/5XTCP 4330
     1X, 22HAUTOMATIC TERMINATION.)                                        TCP 4340
          MARK=1                                                           TCP 4350
          GO TO 725                                                        TCP 4360
  595 CONTINUE                                                             TCP 4370
C                                                                          TCP 4380
C                                                                          TCP 4390
C     THIS SECTION FILLS THE COEFFICIENT MATRIX FOR THE NEWLY ADDED        TCP 4400
C     CONSTRAINT(S) (I.E. TRADEOFF CUTS).  NOTE:  THE LAST CONSTRAINT      TCP 4410
C     IS THE SAME AS THE USABLE DIRECTION CONSTRAINT.  THIS SECTION        TCP 4420
C     ALSO CHECKS TO SEE IF A TRADEOFF CUT EXCEEDS ITS PREVIOUS MAX        TCP 4430
C     VALUE.                                                               TCP 4440
C                                                                          TCP 4450
C                                                                          TCP 4460
          IK=NC+(2*NX)+1                                                   TCP 4470
          IJ=NC+(2*NX)+NACC                                                TCP 4480
          DO 620 I=IK,IJ                                                   TCP 4490
             IL=I-NC-(2*NX)                                                TCP 4500
             DO 605 J=1,NX                                                 TCP 4510
                JJ=NX+J                                                    TCP 4520
                COL(J,I)=0.                                                TCP 4530
                DO 600 IF=1,NCF                                            TCP 4540
                   IK=NX*IF-NX+J                                           TCP 4550
  600           COL(J,I)=COL(J,I)+TRADE(IL,IF)*GF(IK)                      TCP 4560
  605        COL(JJ,I)=-COL(J,I)                                           TCP 4570
             COL(2*NX+1,I)=-1.0                                            TCP 4580
             C(I)=0.                                                       TCP 4590
             DO 610 J=1,NCF                                                TCP 4600
  610        C(I)=C(I)-TRACE(IL,J)*F(J)                                    TCP 4610
             C(I)=C(I)-HOLD(IL)                                            TCP 4620
             CBAR(I)=C(I)                                                  TCP 4630
             IF (-C(I).GT.TCMAX(IL)) TCMAX(IL)=-C(I)                       TCP 4640
             IF (-C(I).GE.0.0) GO TO 620                                   TCP 4650
             WRITE (6,615) I                                               TCP 4660
             WRITE (8,615) I                                               TCP 4670
  615 FORMAT (//1X,I3, 43H ELEMENT OF THE LP COST VECTOR IS POSITIVE.,/5XTCP 4680
     1X, 22HAUTOMATIC TERMINATION.)                                        TCP 4690
             MARK=1                                                        TCP 4700
             GO TO 725                                                     TCP 4710
  620 CONTINUE                                                             TCP 4720
C                                                                          TCP 4730
C                                                                          TCP 4740
C     THIS SECTION DETERMINES THE USABLE DIRECTION H VIA A LINEAR          TCP 4750
C     PROGRAMMING SUBROUTINE.                                              TCP 4760
C                                                                          TCP 4770
C                                                                          TCP 4780
          CALL UDUAL (RM,IFLAG)                                            TCP 4790
          IF (IFLAG) 625,645,635                                           TCP 4800
  625 WRITE (6,630)                                                        TCP 4810
          WRITE (8,630)                                                    TCP 4820
  630 FORMAT (/10X, 38HERROR IN SUBROUTINE UDUAL INPUT DATA.,/10X, 29HCTCP 4830
     1(J) NEED TO BE NON-POSITIVE.,/10X, 10HAUTOMATIC , 12HTERMINATION.)TCP 4840
          MARK=4                                                           TCP 4850
          GO TO 725                                                        TCP 4860
  635 WRITE (6,640)                                                        TCP 4870
```

```
      WRITE (8,640)                                                    TCP 4880
  640 FORMAT (/10X, 42HMINIMUM RATIO RULE FAILED IN DUAL SIMPLEX , 8HROTCP 4890
     1UTINE.,/10X, 22HAUTOMATIC TERMINATION.)                          TCP 4900
      MARK=1                                                           TCP 4910
      GO TO 725                                                        TCP 4920
C                                                                      TCP 4930
C                                                                      TCP 4940
C     CHECK WHETHER RM OR H ARE NEAR ZERO.                             TCP 4950
C                                                                      TCP 4960
C                                                                      TCP 4970
  645 IF (RM.LT.-0.00001) GO TO 650                                    TCP 4980
      GO TO 710                                                        TCP 4990
  650 DO 655 IH=1,NX                                                   TCP 5000
         IF (ABS(H(IH)).GT.0.00001) GO TO 660                          TCP 5010
         IF (IH.EQ.NX) GO TO 710                                       TCP 5020
  655 CONTINUE                                                         TCP 5030
C                                                                      TCP 5040
C                                                                      TCP 5050
C     FINDS THE CENTER OF THE DECISION SPACE.                          TCP 5060
C                                                                      TCP 5070
C                                                                      TCP 5080
  660 CALL GSS (RHO,ERR,X,DIST,RLAMDA)                                 TCP 5090
      ERRN=-ERR                                                        TCP 5100
      IF (DIST.LE.ERRN) GO TO 665                                      TCP 5110
      ERR=ERR/2.                                                       TCP 5120
      GO TO 660                                                        TCP 5130
C                                                                      TCP 5140
C                                                                      TCP 5150
C     THIS SECTION LOCATES THE NEW POINT IN DECISION SPACE             TCP 5160
C     AND EVALUATES IT IN OBJECTIVE SPACE.                             TCP 5170
C                                                                      TCP 5180
C                                                                      TCP 5190
  665 T=1.                                                             TCP 5200
      DO 670 K=1,NX                                                    TCP 5210
         Z(K)=RLAMDA*H(K)                                              TCP 5220
  670 XOLD(K)=X(K)                                                     TCP 5230
      WRITE (6,675) IC                                                 TCP 5240
      WRITE (8,675) IC                                                 TCP 5250
  675 FORMAT (/1X, 48H**********************************************, TCP 5260
     111H***********,//5X,  7HON THE ,I2, 24H ITERATION, THE DECISION, 1TCP 5270
     20H POINT IS:)                                                    TCP 5280
      DO 685 K=1,NX                                                    TCP 5290
         X(K)=XOLD(K)+Z(K)                                             TCP 5300
         WRITE (6,680) K,X(K)                                          TCP 5310
         WRITE (8,680) K,X(K)                                          TCP 5320
  680 FORMAT (15X,  2HX(,I2,  4H) = ,F10.4)                            TCP 5330
  685 CONTINUE                                                         TCP 5340
      CALL FFXNS (X,F)                                                 TCP 5350
      WRITE (6,690) IC                                                 TCP 5360
      WRITE (8,690) IC                                                 TCP 5370
  690 FORMAT (/5X,  7HON THE ,I2, 35H ITERATION, THE OBJECTIVE FUNCTION TCP 5380
     1, 11HVALUES ARE:)                                                TCP 5390
      DO 700 I=1,NOF                                                   TCP 5400
         WRITE (6,695) NAME(I),I,F(I)                                  TCP 5410
         WRITE (8,695) NAME(I),I,F(I)                                  TCP 5420
  695 FORMAT (1X,A10,1CX,  2HF(,I1,  1H),5X,F20.5)                     TCP 5430
         OLDF(NACC+1,I)=F(I)                                           TCP 5440
  700 CONTINUE                                                         TCP 5450
      WRITE (6,705)                                                    TCP 5460
      WRITE (8,705)                                                    TCP 5470
  705 FORMAT (/1X, 47H***********************************************, 1TCP 5480
```

```
     12H************,/)                                                    TCP 5490
      IC=IC+1                                                             TCP 5500
      GO TO 290                                                           TCP 5510
C                                                                         TCP 5520
C                                                                         TCP 5530
C     TERMINATION IN A FINITE NUMBER OF STEPS.                            TCP 5540
C                                                                         TCP 5550
C                                                                         TCP 5560
  710 WRITE (6,715)                                                       TCP 5570
      WRITE (8,715)                                                       TCP 5580
  715 FORMAT (/10X, 43HAUTOMATIC TERMINATION.  THE BEST-COMPROMISE, 18H   TCP 5590
     1HAS BEEN REACHED.)                                                  TCP 5600
      MARK=2                                                              TCP 5610
      GO TO 725                                                           TCP 5620
C                                                                         TCP 5630
C                                                                         TCP 5640
C     SUMMARY REPORT AT TERMINATION.                                      TCP 5650
C                                                                         TCP 5660
C                                                                         TCP 5670
  720 MARK=3                                                              TCP 5680
  725 WRITE (6,730)                                                       TCP 5690
  730 FORMAT (1H1)                                                        TCP 5700
      WRITE (6,735)                                                       TCP 5710
  735 FORMAT (//14X, 28H****************************,/,21X, 14HSUMMARY RTCP 5720
     1EPORT,/14X, 28H****************************,//)                    TCP 5730
      GO TO (740,770,750,760), MARK                                       TCP 5740
  740 WRITE (6,745)                                                       TCP 5750
  745 FORMAT (/5X, 46HTERMINATION DUE TO MINIMUM RATIO RULE FAILURE.)     TCP 5760
      GO TO 780                                                           TCP 5770
  750 IIC=IC-1                                                            TCP 5780
      WRITE (6,755) IIC                                                   TCP 5790
  755 FORMAT (/5X, 51HTERMINATION DUE TO DECISION MAKER PREFERENCE AFTERTCP 5800
     1 ,I2, 12H ITERATIONS.)                                              TCP 5810
      GO TO 780                                                           TCP 5820
  760 WRITE (6,765)                                                       TCP 5830
  765 FORMAT (/5X, 46HTERMINATION DUE TO ERROR IN UCOUAL INPUT DATA.)     TCP 5840
      GO TO 780                                                           TCP 5850
  770 IIC=IC-1                                                            TCP 5860
      WRITE (6,775) IIC                                                   TCP 5870
  775 FORMAT (/5X, 49HTERMINATION DUE TO BEST-COMPROMISE BEING REACHED ,TCP 5880
     1  6HAFTER ,I2, 12H ITERATIONS.)                                     TCP 5890
  780 WRITE (6,785)                                                       TCP 5900
  785 FORMAT (/5X, 40HTHE INITIAL POINT IN DECISION SPACE WAS:)           TCP 5910
      DO 795 I=1,NX                                                       TCP 5920
         WRITE (6,790) I,X1(I)                                            TCP 5930
  790 FORMAT (10X,  2HX(,I2,  4H) = ,F10.4)                               TCP 5940
  795 CONTINUE                                                            TCP 5950
      WRITE (6,800)                                                       TCP 5960
  800 FORMAT (/5X, 43HTHE INITIAL OBJECTIVE FUNCTION VALUES WERE:)        TCP 5970
      DO 810 I=1,NOF                                                      TCP 5980
         WRITE (6,805) NAME(I),I,(LOF(1,I)                                TCP 5990
  805 FORMAT (1X,A10,10X,  2HF(,I1,  1H),5X,F20.5)                        TCP 6000
  810 CONTINUE                                                            TCP 6010
      WRITE (6,815)                                                       TCP 6020
  815 FORMAT (/5X, 37HTHE FINAL POINT IN DECISION SPACE IS:)              TCP 6030
      DO 825 I=1,NX                                                       TCP 6040
         WRITE (6,820) I,X(I)                                             TCP 6050
  820 FORMAT (10X,  2HX(,I2,  4H) = ,F10.4)                               TCP 6060
  825 CONTINUE                                                            TCP 6070
      WRITE (6,830)                                                       TCP 6080
  830 FORMAT (/5X, 40HTHE FINAL OBJECTIVE FUNCTION VALUES ARE:)           TCP 6090
```

```
      DO 840 I=1,NOF                                                    TCP 6100
         WRITE (6,835) NAME(I),I,F(I)                                   TCP 6110
835   FORMAT (1X,A10,10X,  2HF(,I1,  1H),5X,F20.5)                      TCP 6120
840   CONTINUE                                                          TCP 6130
      WRITE (6,845)                                                     TCP 6140
845   FORMAT (/21X, 17HCONSTRAINT VALUES,/,5X,  3HNO.,6X,  5HFINAL,7X,  TCP 6150
     1 7HINITIAL,8X,  3HMIN,10X,  3HMAX)                                TCP 6160
      CALL GFXNS (X,G)                                                  TCP 6170
      FDIST=-G(1)                                                       TCP 6180
      ICON=1                                                            TCP 6190
      DO 850 I=1,NC                                                     TCP 6200
         IF (-G(I).GT.GMAX(I)) GMAX(I)=-G(I)                            TCP 6210
         IF (-G(I).LT.GMIN(I)) GMIN(I)=-G(I)                            TCP 6220
         IF (-G(I).GT.FDIST) GO TO 850                                  TCP 6230
         FDIST=-G(I)                                                    TCP 6240
         ICON=I                                                         TCP 6250
850   CONTINUE                                                          TCP 6260
      DO 860 I=1,NC                                                     TCP 6270
         WG=-G(I)                                                       TCP 6280
         WRITE (6,855) I,WG,GIN(I),GMIN(I),GMAX(I)                      TCP 6290
855   FORMAT (5X,I2,5X,F8.3,5X,F8.3,5X,F8.3,5X,F8.3)                    TCP 6300
860   CONTINUE                                                          TCP 6310
      WRITE (6,865)                                                     TCP 6320
865   FORMAT (/13X, 19HTRADEOFF CLT VALUES,/5X,  3HNO.,4X,  7HCUT NO.,6XTCP 6330
     1,  5HFINAL,9X,  3HMAX)                                            TCP 6340
      NACCM=NACC-1                                                      TCP 6350
      DO 885 K=1,NACCM                                                  TCP 6360
         KK=NC+K                                                        TCP 6370
         TCV=0.                                                         TCP 6380
         IK=NC+(2*NX)+K                                                 TCP 6390
         IL=IK-NC-(2*NX)                                                TCP 6400
         DO 870 J=1,NOF                                                 TCP 6410
870      TCV=TCV+TRADE(IL,J)*F(J)                                       TCP 6420
         TCV=TCV+HOLD(IL)                                               TCP 6430
         IF (TCV.GT.TCMAX(K)) TCMAX(K)=TCV                              TCP 6440
         IF (TCV.GE.FDIST) GO TO 875                                    TCP 6450
         FDIST=TCV                                                      TCP 6460
         ICON=KK                                                        TCP 6470
875      WRITE (6,880) KK,K,TCV,TCMAX(K)                                TCP 6480
880   FORMAT (5X,I2,7X,I2,7X,F8.3,5X,F8.3)                              TCP 6490
885   CONTINUE                                                          TCP 6500
      WRITE (6,890) FDIST,ICON                                          TCP 6510
890   FORMAT (/5X, 31HTHE LOWEST CONSTRAINT VALUE IS ,F8.3/,5X, 25HTHE CTCP 6520
     1ONSTRAINT IS NUMBER ,I2,  1H.)                                    TCP 6530
      WRITE (6,895)                                                     TCP 6540
895   FORMAT (/30X, 44HTRADEOFF SENSITIVITY BASED ON FINAL SOLUTION)    TCP 6550
      DO 950 I=1,NACCM                                                  TCP 6560
         WRITE (6,900) I,CLOF(I,1)                                      TCP 6570
900   FORMAT (/1X, 15HTRADEOFF POINT ,I2, 11H  ( F(1) = ,F10.3,  2H ),//TCP 6580
     1 5X,  8HTRADEOFF,10X,  1HF,12X,  5HGIVEN,14X, 15HALLOW (SUC PTS),18TCP 6590
     2X, 13HALLOW (BC PT),/53X,  3HMIN,13X,  3HMAX,13X,  3HMIN,13X,  3HMTCP 6600
     3AX)                                                               TCP 6610
         DO 945 J=2,NOF                                                 TCP 6620
            SMIN=0.0                                                    TCP 6630
            SMAX=1.E30                                                  TCP 6640
            IK=I+1                                                      TCP 6650
            DO 920 II=IK,NACC                                           TCP 6660
               DF=CLOF(II,J)-CLOF(I,J)                                  TCP 6670
               IF (ABS(DF).LE.0.000001) GO TO 920                       TCP 6680
               SC=0.0                                                   TCP 6690
               DO 905 KM=1,NOF                                          TCP 6700
```

```
                  IF (KM.EQ.J) GO TO 905                                    TCP 6710
                  SC=SC+TRADE(I,KM)*(CLDF(II,KM)-OLDF(I,KM))                TCP 6720
  905          CONTINUE                                                     TCP 6730
               IF (DF.GT.0.0) GO TO 915                                     TCP 6740
               IF (SC.GT.0.0) GO TO 910                                     TCP 6750
               SMIN=-9.                                                     TCP 6760
               SMAX=-9.                                                     TCP 6770
               GO TO 920                                                    TCP 6780
  910          IF (SMIN.LT.0.0) GO TO 920                                   TCP 6790
               TMAX1=SC/(-DF)                                               TCP 6800
               TSMIN=1./TMAX1                                               TCP 6810
               IF (TSMIN.GT.SMIN) SMIN=TSMIN                                TCP 6820
               GO TO 920                                                    TCP 6830
  915          IF (SC.GE.0.0) GO TO 920                                     TCP 6840
               TMIN1=-SC/DF                                                 TCP 6850
               TSMAX=1./TMIN1                                               TCP 6860
               IF (TSMAX.LT.SMAX) SMAX=TSMAX                                TCP 6870
  920        CONTINUE                                                       TCP 6880
             BMIN=0.0                                                       TCP 6890
             BMAX=1.E30                                                     TCP 6900
             DFO=F(J)-OLDF(I,J)                                             TCP 6910
             IF (ABS(DFO).LE.0.000001) GO TO 935                            TCP 6920
             IF (DFO.GT.0.0) GO TO 930                                      TCP 6930
             IF (SC.GT.0.0) GO TO 925                                       TCP 6940
             BMIN=-9.                                                       TCP 6950
             BMAX=-9.                                                       TCP 6960
             GO TO 935                                                      TCP 6970
  925        TMAX=SC/(-DFO)                                                 TCP 6980
             BMIN=1./TMAX                                                   TCP 6990
             GO TO 935                                                      TCP 7000
  930        IF (SC.GE.0.0) GO TO 935                                       TCP 7010
             TMIN=-SC/DFO                                                   TCP 7020
             BMAX=1./TMIN                                                   TCP 7030
  935        CONTINUE                                                       TCP 7040
             DART=1./TRADE(I,J)                                             TCP 7050
             WRITE (6,940) J,OLDF(I,J),DART,SMIN,SMAX,BMIN,BMAX             TCP 7060
  940 FORMAT (8X,  5HJ VS ,I1,5X,F10.3,5X,E11.3,5X,E11.3,5X,E11.3,5X,E11TCP 7070
     1.3,5X,E11.3)                                                          TCP 7080
  945    CONTINUE                                                          TCP 7090
  950 CONTINUE                                                             TCP 7100
  955 STOP                                                                 TCP 7110
C                                                                          TCP 7120
      END                                                                  TCP 7130
      SUBROUTINE UODUAL (RM,IFLAG)                                         UOD   10
C                                                                          UOD   20
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   UOD   30
C                                                                     C    UOD   40
C     THIS SUBROUTINE DETERMINES THE USABLE DIRECTION M               C    UOD   50
C                                                                     C    UOD   60
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   UOD   70
C                                                                          UOD   80
      COMMON /TCOM1/ NX,NOF,NC,NACC                                        UOD   90
      COMMON /TCOM2/ TRADE(20,10),HOLD(20),M(20)                           UOD  100
      COMMON /TCOM3/ CCL(41,121),C(121),CBAR(121)                          UOD  110
      DIMENSION CI(41), IVAR(41), RHS(41)                                  UOD  120
C                                                                          UOD  130
C                                                                          UOD  140
      M=2*NX+1                                                             UOD  150
      N=4*NX+NC+NACC+1                                                     UOD  160
      L=2*NX+NC+NACC+1                                                     UOD  170
      LL=2*NX+NC+NACC                                                      UOD  180
```

```
      EPS=-0.5E-6                                                       UDD  190
      IT=-1                                                             UDD  200
      IFLAG=0                                                           UDD  210
C                                                                       UDD  220
C                                                                       UDD  230
C     INITIALIZE RHS                                                    UDD  240
C                                                                       UDD  250
C                                                                       UDD  260
      MM=M-1                                                            UDD  270
      DO 105 I=1,MM                                                     UDD  280
  105 RHS(I)=0.                                                         UDD  290
      RHS(M)=-1.0                                                       UDD  300
C                                                                       UDD  310
C                                                                       UDD  320
C     TEST WHETHER THE COST COEFFICIENTS HAVE THE NECESSARY             UDD  330
C     NON-POSITIVE FORM.  IF THERE IS A VIOLATION, INDICATE             UDD  340
C     AN ERROR.                                                         UDD  350
C                                                                       UDD  360
C                                                                       UDD  370
      DO 110 J=M,LL                                                     UDD  380
         IF (C(J).GT.0.0) GO TO 115                                     UDD  390
  110 CONTINUE                                                          UDD  400
      GO TO 120                                                         UDD  410
  115 IFLAG=-1                                                          UDD  420
      GO TO 220                                                         UDD  430
C                                                                       UDD  440
C                                                                       UDD  450
C     THIS SECTION FILLS IN THE IDENTITY MATRIX AND COSTS ASSOCIATED    UDD  460
C     WITH THE BASIC VARIABLES.                                         UDD  470
C                                                                       UDD  480
C                                                                       UDD  490
  120 DO 130 I=1,M                                                      UDD  500
         II=2*NX+NC+NACC+I                                              UDD  510
         DO 125 J=L,N                                                   UDD  520
  125    COL(I,J)=0.0                                                   UDD  530
         C(II)=0.0                                                      UDD  540
         CBAR(II)=0.0                                                   UDD  550
         CI(I)=0.0                                                      UDD  560
         IVAR(I)=II                                                     UDD  570
  130 COL(I,II)=1.0                                                     UDD  580
C                                                                       UDD  590
C                                                                       UDD  600
C     THIS SECTION LOCATES THE PIVOT ROW IR.                            UDD  610
C                                                                       UDD  620
C                                                                       UDD  630
  135 IR=0                                                              UDD  640
      DO 140 I=1,M                                                      UDD  650
         IF (RHS(I).GE.EPS) GO TO 140                                   UDD  660
         IR=I                                                           UDD  670
         GO TO 145                                                      UDD  680
  140 CONTINUE                                                          UDD  690
      IF (IR.EQ.0) GO TO 195                                            UDD  700
  145 IT=IT+1                                                           UDD  710
C                                                                       UDD  720
C                                                                       UDD  730
C      THIS SECTION LOCATES THE PIVOT COLUMN IS.                        UDD  740
C                                                                       UDD  750
C                                                                       UDD  760
      IS=0                                                              UDD  770
      THETA=9.E30                                                       UDD  780
      DO 150 J=1,N                                                      UDD  790
```

```
            IF (COL(IR,J).GE.EPS) GO TO 150                                    UDD  800
            RATIO=CBAR(J)/COL(IR,J)                                            UDD  810
            IF (RATIO.GT.THETA) GO TO 150                                      UDD  820
            THETA=RATIO                                                        UDD  830
            IS=J                                                               UDD  840
  150    CONTINUE                                                              UDD  850
         IF (IS.EQ.0) GO TO 190                                                UDD  860
C                                                                              UDD  870
C                                                                              UDD  880
C        UPDATE RHS                                                            UDD  890
C                                                                              UDD  900
C                                                                              UDD  910
         DO 155 I=1,M                                                          UDD  920
            IF (I.EQ.IR) GO TO 155                                             UDD  930
            RHS(I)=RHS(I)-(COL(I,IS)/COL(IR,IS))*RHS(IR)                       UDD  940
C                                                                              UDD  950
C                                                                              UDD  960
C        CHECK FOR ROUND-OFF ERROR CLOSE TO ZERO AND CORRECT IF NECESSARY.     UDD  970
C                                                                              UDD  980
C                                                                              UDD  990
            IF (ABS(RHS(I)).LT.1.0E-10) RHS(I)=0.                              UDD 1000
  155    CONTINUE                                                              UDD 1010
C                                                                              UDD 1020
C                                                                              UDD 1030
C        UPDATE RHS(IR)                                                        UDD 1040
C                                                                              UDD 1050
C                                                                              UDD 1060
         RHS(IR)=RHS(IR)/COL(IR,IS)                                            UDD 1070
C                                                                              UDD 1080
C                                                                              UDD 1090
C        UPDATE CBAR ROW FOR THE NEXT ITERATION.                               UDD 1100
C                                                                              UDD 1110
C                                                                              UDD 1120
         DO 160 J=1,N                                                          UDD 1130
            CBAR(J)=CBAR(J)-THETA*COL(IR,J)                                    UDD 1140
            IF (ABS(CBAR(J)).LT.1.0E-10) CBAR(J)=0.                            UDD 1150
  160    CONTINUE                                                              UDD 1160
         CBAR(IS)=0.0                                                          UDD 1170
C                                                                              UDD 1180
C                                                                              UDD 1190
C        THE NEW COL(I,J) MATRIX IS CALCULATED.                                UDD 1200
C                                                                              UDD 1210
C                                                                              UDD 1220
         DO 170 J=1,N                                                          UDD 1230
            DO 165 I=1,M                                                       UDD 1240
               IF (I.EQ.IR) GO TO 165                                          UDD 1250
               IF (J.EQ.IS) GO TO 170                                          UDD 1260
               COL(I,J)=COL(I,J)-(COL(I,IS)/COL(IR,IS))*COL(IR,J)              UDD 1270
               IF (ABS(COL(I,J)).LT.1.0E-10) COL(I,J)=0.                       UDD 1280
  165       CONTINUE                                                           UDD 1290
  170    CONTINUE                                                              UDD 1300
         DO 180 J=1,N                                                          UDD 1310
            IF (J.EQ.IS) GO TO 175                                             UDD 1320
            COL(IR,J)=COL(IR,J)/COL(IR,IS)                                     UDD 1330
  175       CONTINUE                                                           UDD 1340
  180    CONTINUE                                                              UDD 1350
         DO 185 I=1,M                                                          UDD 1360
            COL(I,IS)=0.                                                       UDD 1370
  185    CONTINUE                                                              UDD 1380
         COL(IR,IS)=1.                                                         UDD 1390
C                                                                              UDD 1400
```

```
C                                                                       UDD 1410
C     UPDATES INDEX ARRAY OF BASIC VARIABLES IVAR(I), BASIC VARIABLE    UDD 1420
C     COSTS CI(I), AND THE TOLERANCE ERROR.  IT RETURNS TO CONTINUE     UDD 1430
C     THE DUAL SIMPLEX PROCEDURE.                                       UDD 1440
C                                                                       UDD 1450
C                                                                       UDD 1460
      CI(IR)=C(IS)                                                      UDD 1470
      IVAR(IR)=IS                                                       UDD 1480
      EPS=EPS-.5E-6                                                     UDD 1490
      GO TO 135                                                         UDD 1500
C                                                                       UDD 1510
C                                                                       UDD 1520
C     MINIMUM RATIO RULE FAILED.  PROBLEM IS INFEASIBLE.                UDD 1530
C                                                                       UDD 1540
C                                                                       UDD 1550
  190 IFLAG=1                                                           UDD 1560
      GO TO 220                                                         UDD 1570
C                                                                       UDD 1580
C                                                                       UDD 1590
C     DETERMINES PRIMAL SOLUTION AND FILLS H VECTOR.                    UDD 1600
C                                                                       UDD 1610
C                                                                       UDD 1620
  195 VALUE=0.0                                                         UDD 1630
      DO 200 I=1,M                                                      UDD 1640
  200 VALUE=VALUE+RHS(I)*CI(I)                                          UDD 1650
      RM=VALUE                                                          UDD 1660
      DO 210 K=1,M                                                      UDD 1670
         KK=2*NX+NC+NACC+K                                              UDD 1680
         RHS(K)=0.                                                      UDD 1690
         DO 205 I=1,M                                                   UDD 1700
            RHS(K)=RHS(K)+CI(I)*COL(I,KK)                               UDD 1710
  205    CONTINUE                                                       UDD 1720
         IF (ABS(RHS(K)).LT.1.0E-10) RHS(K)=0.                          UDD 1730
  210 CONTINUE                                                          UDD 1740
      DO 215 I=1,NX                                                     UDD 1750
         II=NX+I                                                        UDD 1760
  215 H(I)=RHS(I)-RHS(II)                                               UDD 1770
  220 RETURN                                                            UDD 1780
C                                                                       UDD 1790
      END                                                               UDD 1800
      SUBROUTINE GSS (RHO,ERR,X,DIST,FLAMDA)                            GSS   10
C                                                                       GSS   20
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC  GSS   30
C                                                                C      GSS   40
C     GOLDEN SECTION SEARCH SUBROUTINE                           C      GSS   50
C                                                                C      GSS   60
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC  GSS   70
C                                                                       GSS   80
      DIMENSION X(20)                                                   GSS   90
C                                                                       GSS  100
C                                                                       GSS  110
C     THIS SECTION COMPUTES AN INTERVAL (RIM1,RJ) CONTAINING A          GSS  120
C     MINIMIZING POINT.                                                 GSS  130
C                                                                       GSS  140
C                                                                       GSS  150
      RIM1=0.                                                           GSS  160
      RI=0.                                                             GSS  170
      CALL EVALD (X,RI,FI,ICON)                                         GSS  180
      FIM1=FI                                                           GSS  190
  105 RJ=RI+RHO                                                         GSS  200
      CALL EVALD (X,RJ,FJ,ICON)                                         GSS  210
```

```
      IF (FJ.GE.FI) GO TO 110                                          GSS  220
      FIM1=FI                                                          GSS  230
      FI=FJ                                                            GSS  240
      RIM1=RI                                                          GSS  250
      RI=RJ                                                            GSS  260
      GO TO 105                                                        GSS  270
  110 CONTINUE                                                         GSS  280
C                                                                      GSS  290
CC                                                                     GSS  300
C     A MINIMIZING POINT EXISTS BETWEEN RIM1 AND RJ.                   GSS  310
CC                                                                     GSS  320
C                                                                      GSS  330
      A=RIM1                                                           GSS  340
      B=RJ                                                             GSS  350
      FA=FIM1                                                          GSS  360
      FB=FJ                                                            GSS  370
C                                                                      GSS  380
CC                                                                     GSS  390
C     THIS SECTION REDUCES THE LENGTH OF THE INTERVAL CONTAINING THE    GSS  400
C     MINIMIZING POINT.                                                GSS  410
CC                                                                     GSS  420
C                                                                      GSS  430
      R=0.618033                                                       GSS  440
      RSQ=R*R                                                          GSS  450
      XW=B-A                                                           GSS  460
      XL=A+RSQ*XW                                                      GSS  470
      XR=A+R*XW                                                        GSS  480
      CALL EVALO (X,XL,FL,ICON)                                        GSS  490
      CALL EVALO (X,XR,FR,ICON)                                        GSS  500
  115 IF (FR.LT.FL) GO TO 120                                          GSS  510
C                                                                      GSS  520
CC                                                                     GSS  530
C     TEST THE REMAINING INTERVAL (A,XR).  DISCARD ALL BEYOND          GSS  540
C     X-RIGHT, AND COMPUTE A NEW X-LEFT.                               GSS  550
CC                                                                     GSS  560
C                                                                      GSS  570
      REGION=XR-A                                                      GSS  580
      IF (REGION.LE.ERR) GO TO 125                                     GSS  590
      B=XR                                                             GSS  600
      FB=FR                                                            GSS  610
      XW=XR-A                                                          GSS  620
      XR=XL                                                            GSS  630
      FR=FL                                                            GSS  640
      XL=A+RSQ*XW                                                      GSS  650
      CALL EVALO (X,XL,FL,ICON)                                        GSS  660
      GO TO 115                                                        GSS  670
C                                                                      GSS  680
CC                                                                     GSS  690
C     TEST THE REMAINING INTERVAL (XL,B).  DISCARD ALL BELOW X-LEFT,   GSS  700
C     AND COMPUTE A NEW X-RIGHT.                                       GSS  710
CC                                                                     GSS  720
C                                                                      GSS  730
  120 REGION=B-XL                                                      GSS  740
      IF (REGION.LE.ERR) GO TO 130                                     GSS  750
      A=XL                                                             GSS  760
      FA=FL                                                            GSS  770
      XW=B-XL                                                          GSS  780
      XL=XR                                                            GSS  790
      FL=FR                                                            GSS  800
      XR=A+R*XW                                                        GSS  810
      CALL EVALO (X,XR,FR,ICON)                                        GSS  820
```

```
      GO TO 115                                                          GSS  830
C                                                                        GSS  840
C                                                                        GSS  850
C     THE SOLUTION LIES BETWEEN A AND XR.                                GSS  860
C                                                                        GSS  870
C                                                                        GSS  880
  125 XRT=XR                                                             GSS  890
      XLF=A                                                              GSS  900
      GO TO 135                                                          GSS  910
C                                                                        GSS  920
C                                                                        GSS  930
C     THE SOLUTION LIES BETWEEN XL AND B.                                GSS  940
C                                                                        GSS  950
C                                                                        GSS  960
  130 XRT=B                                                              GSS  970
      XLF=XL                                                             GSS  980
C                                                                        GSS  990
C                                                                        GSS 1000
C     PRINTOUT                                                           GSS 1010
C                                                                        GSS 1020
C                                                                        GSS 1030
  135 RLAMDA=(XRT+XLF)*0.5                                               GSS 1040
      CALL EVALC (X,RLAMDA,DIST,ICON)                                    GSS 1050
      PDIST=-DIST                                                        GSS 1060
      WRITE (6,140) PDIST,ICON                                           GSS 1070
      WRITE (8,140) PDIST,ICON                                           GSS 1080
  140 FORMAT (/10X, 31HTHE LOWEST CONSTRAINT VALUE IS ,F10.3/,10X, 25HTHGSS 1090
     1E CONSTRAINT IS NUMBER ,I2,  1H.)                                  GSS 1100
      RETURN                                                             GSS 1110
C                                                                        GSS 1120
      END                                                                GSS 1130
      SUBROUTINE EVALC (X,C,CVM,ICON)                                    EVA   10
C                                                                        EVA   20
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   EVA   30
C                                                                  C     EVA   40
C     THIS SUBROUTINE EVALUATES THE MINIMUM DISTANCE TO THE        C     EVA   50
C                          BOUNDARY.                               C     EVA   60
C                                                                  C     EVA   70
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   EVA   80
C                                                                        EVA   90
      COMMON /TCOM1/ NX,NCF,NC,NACC                                      EVA  100
      COMMON /TCOM2/ TRADE(20,10),HCLO(20),H(20)                         EVA  110
      DIMENSION XNEW(20), X(20), F(10), G(20)                            EVA  120
C                                                                        EVA  130
C                                                                        EVA  140
      DO 105 I=1,NX                                                      EVA  150
  105 XNEW(I)=X(I)+C*H(I)                                                EVA  160
      CALL FFXNS (XNEW,F)                                                EVA  170
      CALL GFXNS (XNEW,G)                                                EVA  180
C                                                                        EVA  190
C                                                                        EVA  200
C     FINDS THE MINIMUM DISTANCE TO ALL THE ORIGINAL CONSTRAINTS.        EVA  210
C                                                                        EVA  220
C                                                                        EVA  230
      ICON=1                                                             EVA  240
      CVM=G(1)                                                           EVA  250
      I=2                                                                EVA  260
  110 IF (G(I).LE.CVM) GO TO 115                                         EVA  270
      ICON=I                                                             EVA  280
      CVM=G(I)                                                           EVA  290
  115 I=I+1                                                              EVA  300
```

```
      IF (I.LE.NC) GO TO 110                                          EVA  310
C                                                                     EVA  320
C                                                                     EVA  330
C     FINDS THE MINIMUM DISTANCE TO ALL THE CONSTRAINTS.              EVA  340
C                                                                     EVA  350
C                                                                     EVA  360
      DO 125 K=1,NACC                                                 EVA  370
         GCNX=0.                                                      EVA  380
         DO 120 KK=1,NCF                                              EVA  390
            GCNX=GCNX-TRADE(K,KK)*F(KK)                               EVA  400
  120    CONTINUE                                                     EVA  410
         GCNX=GCNX-HOLD(K)                                            EVA  420
         IF (GCNX.LE.CVM) GO TO 125                                   EVA  430
         ICON=NC+K                                                    EVA  440
         CVM=GCNX                                                     EVA  450
  125 CONTINUE                                                        EVA  460
      RETURN                                                          EVA  470
C                                                                     EVA  480
      END                                                             EVA  490
      SUBROUTINE HOLDC (NCN,F)                                        HCL   10
C                                                                     HCL   20
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC HCL   30
C                                                                   C HCL   40
C     THIS SUBROUTINE DETERMINES THE CONSTANT TERM ASSOCIATED       C HCL   50
C     WITH EACH NEW CONSTRAINT (I.E. TRADEOFF CUT).                 C HCL   60
C                                                                   C HCL   70
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC HCL   80
C                                                                     HCL   90
      COMMON /TCOM1/ NX,NCF,NC,NACC                                   HCL  100
      COMMON /TCOM2/ TRADE(20,10),HOLD(20),H(20)                      HCL  110
      DIMENSION F(10)                                                 HCL  120
C                                                                     HCL  130
C                                                                     HCL  140
      HOLD(NCN)=0.                                                    HCL  150
      DO 105 K=1,NOF                                                  HCL  160
  105 HOLD(NCN)=HOLD(NCN)-TRADE(NCN,K)*F(K)                           HCL  170
      RETURN                                                          HCL  180
C                                                                     HCL  190
      END                                                             HCL  200
      SUBROUTINE FFXNS (X,F)                                          FFX   10
      DIMENSION X(2), F(2)                                            FFX   20
      F(1)=-(X(1)**2)/10.+1.6*X(1)                                    FFX   30
      F(2)=-(X(2)**2)/10.+X(2)                                        FFX   40
      RETURN                                                          FFX   50
C                                                                     FFX   60
      END                                                             FFX   70
      SUBROUTINE GFXNS (X,G)                                          GFX   10
      DIMENSION X(2), G(4)                                            GFX   20
      G(1)=(X(1)**2)-6.*X(1)+4.*X(2)-11.                              GFX   30
      G(2)=EXP(X(1)-3.)-X(1)*X(2)+3.*X(2)-1.                          GFX   40
      G(3)=-X(1)                                                      GFX   50
      G(4)=-X(2)                                                      GFX   60
      RETURN                                                          GFX   70
C                                                                     GFX   80
      END                                                             GFX   90
      SUBROUTINE FGPAC (X,GF)                                         FGR   10
      DIMENSION X(2), GF(4)                                           FGR   20
      GF(1)=-0.2*X(1)+1.6                                             FGR   30
      GF(2)=0.                                                        FGR   40
      GF(3)=0.                                                        FGR   50
      GF(4)=-0.2*X(2)+1.                                              FGR   60
```

```
      RETURN                                                        FGR   70
C                                                                   FGR   80
      END                                                           FGR   90
      SUBROUTINE GGRAC (X,GG)                                       GGR   10
      DIMENSION X(2), GG(8)                                         GGR   20
      GG(1)=2.*X(1)-6.                                              GGR   30
      GG(2)=4.                                                      GGR   40
      GG(3)=EXP(X(1)-3.)-X(2)                                       GGR   50
      GG(4)=-X(1)**3.                                               GGR   60
      GG(5)=-1.                                                     GGR   70
      GG(6)=0.                                                      GGR   80
      GG(7)=0.                                                      GGR   90
      GG(8)=-1.                                                     GGR  100
      RETURN                                                        GGR  110
C                                                                   GGR  120
      END                                                           GGR  130
*EOR
*EOF
```

Lilly Hall of Life Sciences
Purdue University
West Lafayette, Indiana 47907

BULK RATE
Non-profit Organization
U. S. Postage
PAID
Permit No. 121
Lafayette, Indiana